A bioinformatics tool to detect physical gene clusters in functional genomics data

A thesis submitted in partial fulfilment of the requirements for the degree of

Master of Science in Biochemistry

from

STELLENBOSCH UNIVERSITY, SOUTH AFRICA

DEPARTMENT OF BIOCHEMISTRY

Faculty of Science

by

Louis Conradie

Promotor: Prof Hugh Patterton

March 2021

ABSTRACT

Many studies have investigated the biological effects of the external environment on the genetic expression of a living cell. This particular study looks at gene clusters that are switched on at a series of discrete time points when *Saccharomyces cerevisiae* comes out of stationary phase, when the cell is fed a carbon source after a starvation period. To achieve this, *Pyxis2* was developed, a bioinformatics tool that is able to detect gene clusters in functional genomics data from any organism. The program detects physical clusters of genes that share a defined functional genomic property, such as transcriptional activity, that may be interpreted in terms of a biological functionality. *Pyxis2* provides several options to the user that may be adjusted for appropriate levels of sensitivity. Pyxis2 is available at https://github.com/Louis-Conradie/Pyxis2_classes_2020.

Following the identification of gene clusters that are induced during the exit of stationary phase in yeast, the study is extended to also investigate the biological relationship between the identified genes and possible regulatory mechanisms. These mechanisms may include transcriptional activators and co-activators, epigenetic modifications such as histone H3K9ac acetylation, or the effect of the domain wide change of the spatial structure of chromatin. I show that the mega-Dalton transcriptional activation complex SAGA is associated with some of the transcriptionally induced clusters at early times during stationary phase exit, and that acetylation of lysine 9 of histone H3 is not a detectable property of active clusters. I finally show some association between spatial proximity of parts of the genome and cluster gene induction but find this association in cycling cells as opposed to in cells re-engaging passage through the cell-cycle. I finally discuss the implications of my findings in the context of the current literature and identify future avenues of enquiry.

OPSOMMING

Verskeie studies het die biologiese effek van die eksterne omgewing op die genetiese uitdrukking in ‘n lewendige sel ondersoek. Hierdie spesifieke studie kyk na geen klusters wat na diskrete tyd periodes aangeskakel word wanneer *Saccharomyces cerevisiae* uit stasionêre fase uitkom nadat die sel na ‘n verhongering periode met ‘n koolstof bron gevoer is. Om dit te bereik is Pyxis2 ontwikkel, ‘n bioinformatika toepassing wat die vermoë het om geen klusters in funksionele genomiese data van enige organisme te identifiseer. Die program identifiseer fisiese klusters van gene wat ‘n funksionele genomika eienskap deel. Soos transkripsionêle aktiwiteit, wat binne ‘n breër biologiese funksionaliteit geïnterpreteer mag word. Pyxis2 verskaf verskeie opsies aan die gebruiker wat verstel mag word vir geskikte vlakke van sensitiwiteit. Pyxis2 is beskikbaar by https://github.com/Louis-Conradie/Pyxis2_classes_2020.

Na die identifisering van geen klusters wat die geïnduseer is gedurende die verlating van stasionêre fase in gis, is die studie uitgebrei om ook die biologiese verwantskap tussen die geïdentifiseerde gene en moontlike regulatoriese meganismes te ondersoek. Hierdie meganismes kan transkripsionêle aktiveerders en ko-aktiveerders insluit, epigenetiese modifikasies soos H3K9ac, of die effek van die domein wye verandering in die ruimtelike struktuur van chromatien. Ek wys dat die mega-Dalton transkripsionêle aktiveerder SAGA kompleks met sommige van die aktiewe klusters tydens vroeë tye met die verlaat van stasionêre fase geassosieerd is, en dat die asetilering van lisien 9 van histoon H3 nie ‘n waarneembare eienskap van aktiewe klusters is nie. Ek toon finaal ‘n assosiasie tussen die ruimtelike nabyheid van gedeeltes van die genoom en kluster geen induksie, maar vind hierdie assosiasie in selle wat deur die sel siklus roteer eerder is in selle wat die sel siklus betree. Ek bespreek finaal die implikasies van my bevindinge in die konteks van die huidige literatuur, en identifiseer moontlike toekomstige paaie van ondersoek.

DECLARATION

I, Louis Conradie, declare that this thesis submitted to Stellenbosch University is wholly my own work and has not been submitted before for any degree at this or any other institution.

Signature: …… Date: ……

Supervisor:

Professor Hugh-George Patterton (PhD)
Director of Centre for Bioinformatics and Computational Biology (CBCB)
Department of Biochemistry
Stellenbosch University
P.O. Box, Stellenbosch. 7600, South Africa

DEDICATION

To my friends and family who provided support and also God Almighty for the strength to persevere during this time of my life.

ACKNOWLEDGEMENTS

I would like to acknowledge the following people for their support during this study:

My supervisor Professor Hugh Patterton for his support and encouragement during my time at Stellenbosch University. He ensured that I was well supported during my studies, and was patient when I did not fully grasp all the concepts surrounding my project. His belief in me to complete my studies is much appreciated even with me coming from a biochemistry background. He inspired me to be the best version of myself and his knowledge and hard work is truly exceptional and something to aspire too.

The CBCB team for their welcoming attitude and making me feel comfortable and being so supportive of me. Special mention to Dr. Riaan de Witt who provided unending support and assistance during my study and also Mrs Olivia Van Wyk for making sure the administration surrounding my project ran smoothly as well as all her wonderful gestures during my time at the office.

To my friends and family, I have learnt so much from you all during the years and all your support during this time. Thank you so much.

TABLES OF FIGURES

Figure 3.10. Phylogenetic tree related to enriched cluster and that are found in closely related species

LIST OF TABLES

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviations	Description
BH	Benjamini Hochberg
BLAST	Basic Local Alignment Search Tool
CLI	Command Line Interface
CSV	Comma Separate Value
DNA	Deoxynucleic Acid
DSIF	DRB Sensitivity Factor
DRB	5,6-dichloro-l3-ribofuranosylbenzimidazole
GFF3	General Feature Format 3
GO	Gene Ontology
GTF	General Transcription Factor
FDR	False Discovery Rate
HAT	Histone Acetyltransferase
HDAC	Histone Deactylatetransferase
IGB	*Integrated Genome Browser*
LCR	Locus Control Region
mRNA	Messenger RNA
NELF	Negative Elongation Factor
NFR	Nucleosome Free Region
ORF	Open Reading Frame
PCR	Polymerase Chain Reaction

PIC	Pre-initiation Complex
rRNA	Ribosomal RNA
SAGA	Spt-Ada-Gcn5-Acetyltransferase
TAF	TBP Associating Factors
TBP	TATA-Binding Protein
tRNA	Transfer RNA
RNA	Ribonucleic Acid
WGD	Whole Genome Duplication

CHAPTER 1

Literature Review

1.1. Introduction

Organisms have the inherent ability to rapidly adapt to external environmental factors that require the adjustment of intra-cellular conditions and changes in the equilibria of biochemical reactions. In prokaryotes, the interconnected nature of the conditions and reactions are often regulated by organising genes with similar functions into groups, called operons. This, in its simplest implementation, allows the induction of all enzymes required to activate a specific metabolic pathway, using a single regulatory switch that controls all the genes encoding the required enzymes.

In eukaryotes this clustering of genes is less common, although some clustering has been observed for β-globin genes in the β-globin locus control region (1), as well as in muscle development genes (2) or the DAL loci in yeast (3). The local, co-regulated groups of genes in eukaryotes are referred to as physical gene clusters, and it was suggested that these groups are regulated by structural changes of the chromatin, often associated with epigenetic modifications, which consequently lead to the coherent regulation of genes groups.

1.2. Gene expression and transcriptional regulation in eukaryotes

The expression of protein coding genes plays a direct role in cellular metabolism, and are primarily regulated at transcriptional level. There are about 25 000 protein coding genes in the human genome, and these genes need to be regulated co-ordinately in time, depending on the dynamics of cellular conditions. (4) This is only one example which emphasizes the complexity of gene regulation. Studies on prokaryotic organisms such as bacteria has provided much insight into gene expression regulatory mechanisms. The classic paper published in 1961 by Jacob and Monod, which explains the regulated gene expression of the lac operon in *Escherichia coli* (4), is a case in point. This discovery led to significant insight into more complex regulatory mechanisms in eukaryotes.

Sam Weiss discovered the first RNA polymerase in 1959 in rat liver cell nuclei, and that discovery paved the way for identifying and understanding the three enzymes, RNA polymerase I, II and III, responsible for transcribing functional units in eukaryotic genomes. Robert Roeder discovered the three different enzymes, RNA polymerase I, II and III, with their different structural domains and enzymatic functions in relation to RNA synthesis. This discovery sparked a big interest in the mechanisms of eukaryotic gene expression, and

also showed how transcription differed in prokaryotes and eukaryotes. (4) Furthermore, it was shown that each RNA polymerase in eukaryotes were responsible for transcribing different types of functional units. For example, RNA polymerase I, is primarily located in the cell nucleolus and transcribes mainly ribosomal protein genes, Pol II is responsible for the transcription of all protein coding genes and a few ribosomal genes, and Pol III is responsible for the transcription of the 5S RNA, U6 snRNA and tRNA genes. (5, 6)

The biochemical mechanism of gene regulation was initially studied by the systematic assembly of active transcription complexes *in vitro*. (7) These assemblies were initially performed on free DNA, but it was subsequently appreciated that the true substrate for RNA polymerase *in situ*, was DNA packaged into chromatin. Chromatin is formed by the repetitive spooling of the DNA molecule onto nucleosomes. Each nucleosome contains about 168 bp of duplex DNA spooled as two negative superhelical turns onto an octamer of histones, composed of two copies of each of histones H2A, H3B, H3 and H4. Depending on local functional requirements, minor isoforms of some of the histones, such as H2A.Z, may be present in the nucleosome, conferring specific structural properties to the nucleosome. The N-terminal tail of some of the histones may also display a reversible post-translational modification (PTM), including acetylation, methylation and phosphorylation. These PTMs generally act as molecular flags, recognised by diverse regulatory proteins, that are recruited to the tagged region of chromatin to perform specific functions such as chromatin remodelling or histone isotype swapping or assembly of a pre-initiation complex.

It was shown that nucleosomes inhibited the process of transcription. (8) This is most easily understood by considering an RNA Pol II enzyme proceeding along a propagating transcription bubble, synthesizing the RNA molecule complementary to the DNA coding strand, and encountering a nucleosome. The DNA is tightly associated with the surface of the nucleosome, and it is energetically unfavourable to dissociate the DNA duplex from this surface, a structural transition that is required to propagate the transcription bubble and proceed with the transcription process.

It was shown in *Saccharomyces cerevisiae* that the absence of histone H4 led to the upregulation of transcription. (9) In the 1980s is was found that transcription factors that were “sequence specific”, bind to promotors and enhancers, and activate gene expression. (10) It was then thought that transcription factors influence the frequency of transcription initiation, and that chromatin did not have any role in gene regulation. (8) However, studies challenged that claim when it was found that the GAGA factor, a transcription factor in *Drosophila*, is not responsible for the direct upregulation of gene expression but worked against a gene repressive state. (8) Histone H1, a fifth “linker” histone that binds to the outside of the nucleosome at the site of DNA entry and exit, and is absolutely required for full chromatin compaction, was found to be a repressor, which indicated that it normally played a role in repression of gene expression. (8) It was also found that transcription factor *Sp1* antagonises the H1 repression mechanism (8) It was furthermore found that the repressor histone H1, found in most transcription “crude extracts”, is needed for transcription, since little

transcription was observed with only DNA templates and transcription factors. (11) This highlighted the importance of histones in gene regulation and became known as the "anti-repression hypothesis".

Gene expression in eukaryotes is essential for all cellular metabolism, and this process is accomplished by the functioning of three RNA polymerases. These enzymes, is isolation, are unable to start transcription on promotor elements. Transcription of the 5S RNA genes in *Xenopus* oocyte chromatin with pol III illustrated the requirement of additional factors that must also associates with the chromatin template to allow transcription. (12) These factors were identified through cellular fractionation of mammalian cells, and became known as general initiation factors or basal factors, specifically TFIIB and TFIIC for Pol III and TFIID for Pol II, but a study by Maramatsu *et al* have also shown that there are factors required for Pol I driven transcription. (13, 14) This discovery led to the description of the pre-initiation complex (PIC) for Pol I and II, which is the co-ordinated arrangement of the general initiation factors such as TFIIC and TFIIB. (4) How does the binding of the basal transcription factors and assembly of the PIC work, and how does this fit into gene organization in eukaryotes?

The cellular transcription of genes in eukaryotes are driven by the three polymerases, RNA Pol I, II and III. These three enzymes differ in the type of genes they transcribe. Firstly, Pol I is responsible for the transcription of the rRNA genes, which is responsible for up to 60% of transcriptional activity in the cell. (15) The importance of this polymerase is highlighted through the production of 47S pre-RNA, which is further processed into 18S, 5,8S and 28S rRNA, which is crucial for assembly of ribosomes. (16) Ribosomes plays a pivotal role in protein synthesis which is important for cellular growth and division. (17) The initiation of gene transcription by Pol I in mammals rely on selectivity factor 1 (SL1) which forms part of the TATA-binding protein, and four TBP-associated factors: TAF_I110, TAF_I63, TAF_I48 and TAF_I41. (17) It was also found that an additional TAF, TAF12, is also crucial for the SL1 complex in mammals. (17) SL1 is essential for TBP to identify and associate with the promotor element in the rDNA "repeat". (17) SL1 has also been shown to be important for the recruitment of TAF1B, which is biochemically related to TFIIB, and the Brf proteins, which are important for Pol II and Pol III transcription. (18) SL1 also plays a part in causing the actively transcribed rRNA genes into a "hypomethylated" state through TAF12 recruitment of GADD45a and many other components of the nucleotide excision repair (NER) machinery. (19)

RNA polymerase II is responsible for the transcription of most protein coding genes and also some small RNAs. Similarly, to Pol I and III, Pol II cannot initiate transcription without the support of transcription factors together with transcription activators and co-activators. (20) An interesting observation was made on the influence of RNA polymerase II on ribosome production in living cells. RNA Pol I and III are not the only known enzymes that influence the transcription of genes related to rRNA. The study showed that RNA pol II in human nucleoli functioned in close proximity to the rRNA genes, and directly influenced the degree of their expression. Pol II and senataxin, an enzyme that is associated with neurodegeneration, forms a triplex nucleic

acid structure that is known as an R-loop at intergenic spaces that is close to rRNA genes in the nucleolus. This structure inhibits pol I from transcribing intergenic, non-coding RNA that can interrupt the organization of the nucleolus and flow of rRNA production. This process can be reversed if pol II function stops, or if senataxin levels decrease. (21)

RNA polymerase III transcribes small RNAs that are not translated by other polymerases, such as tRNA, 5S RNA and U6 snRNA. It was found that Maf1 is a repressor of Pol III through the inhibition of Pol II to interact with the transcription initiation complex that is already formed. (22) Growing cells utilise a significant amount of energy to produce tRNAs. (5) When the cell enters a starvation condition, Maf1 inhibits Pol III from producing tRNAs. (22) Rapamycin (TOR) kinase controls Maf1, and through various signals, is important for cellular growth and development. (5, 23) A study where Maf1 was deleted, demonstrated overproduction of pre-tRNA, but did not cause mature tRNA overproduction. (24)

The three RNA polymerases transcribe genes differently because they have different chemical structures. However, each of these polymerases have an active site where the polymerisation reaction takes place for transcription to occur. (6) For a quick overview of the transcription process we will look into transcription as performed by RNA polymerase II, since it is responsible for the transcription of most genes.

The process starts with specific factors in close proximity to the transcription initiation site. These factors can influence transcription through changing chromatin structure, or interaction with factors of the transcription complexes. (25) Both these changes lead to the transcription complex interacting with a core promotor, where it forms a pre-initiation complex (PIC), but not in a state able to directly initiate transcription. The polymerase enzyme needs to recognise the core promotor with the assistance of the transcription complex. There are many core elements found in the core promotor, such as the TATA box (where the TBP binds), BRE (where TFIIB binds), Inr and DPE. (25) There are also many transcription factors that can stay at a promotor in the so-called “scaffold complex”. (26) This complex remains at genes that have been transcribed and makes the next round of transcription faster, decreasing the time it takes to reassemble a new PIC for a next round of transcription. Certain “transcription activation domains” can maintain these scaffold complexes *in vitro*. (25) It was also thought that most Pol II core promotors contain a TATA element, but it was shown that only about 30% of Pol II transcribed genes in *Drosophila* contained a TATA element. The TATA-less promotors in humans and *Drosophila* contain a mixture of Inr and DPE elements (16, 27) It has been shown in many studies that a mutation of the TATA box in promotors can cause transcription to decrease. (26, 28) A study of the HIS4 promotor in yeast, where the TATA box was mutated to a GC rich sequence, showed the complete abrogation of transcription initiation, emphasizing the importance of the TATA box in transcription. (25)

1.3. Regulation mechanisms of gene expression in eukaryotes

Eukaryotic gene expression is regulated at three levels: the level of gene sequence (the interplay of factors that influence transcription and regulatory sequence units), the level of chromatin (the state in which the chromatin exerts itself through switching between decondensed, active and condensed, inactive state) and lastly at the nuclear level (the effects of the different compartments within the nucleus that affects the chromatin structure and consequently gene expression). (29)

1.3.1. Regulation at the level of gene sequence

Regulation at sequence level entails the interplay of many factors that influence transcription. Firstly, the expression of genes starts when remodellers open the chromatin structure which leads to the recruitment of Pol II with the assistance of general transcription factors to a promotor-proximal pause site. (30) Studies using high-resolution mapping of Pol II by ChIP-seq and ChIP-seq exo experiments indicated that the majority of Pol II is at a pause site, and not at the site of transcription initiation. (31, 32) This indicates that transcription initiation is not the main point of regulation. (30)

What factors play a role in this pause step before initiation of RNA synthesis? 5,6-Dichloro-l3-ribofuranosylbenzimidazole (DRB), which is a purine analogue, is a kinase inhibitor. It was shown that DRB inhibits mRNA production in cells but had no effect on inhibiting transcription *in vitro* with RNA polymerase and transcription factors. (33) This indicated that there are more factors that play a role at this level of gene regulation than is present in the *in vitro* reconstituted system. Two factors associated with DRB, DRB-sensitivity inducing factor (DSIF) and negative elongation factor (NELF), has been shown to cause RNA polymerase to pause at a promotor pause site. (34, 35) Another factor is positive elongation factor (P-TEFb), which promotes transcription *in vitro* through its kinase activity by phosphorylating the complex at the pause site (RNA polymerase, DSIF and NELF) which consequently causes elongation. (31)

Many studies started to take advantage of the recent developments in genome wide analysis techniques, such as the study by Adelman *et al,* where, using a ChIP-chip method, it was demonstrated that Pol II assembled at multiple promotor sites of genes. (36) A study also showed that promotors in humans and *Drosophila* cells had this “proximal-promotor pause” sites. (37) Almost all genes have been shown to go through this pause step. (30) How does this process happen?

The chromatin structure is opened by certain transcription factors and chromatin remodellers, which causes the RNA polymerase and its associated factor to assemble at its respective promotor. DSIF and NELF associates with the complex, which causes it pause at a certain site on “proximal-promotor pause site”. The p-TEFb factor gets recruited by certain transcription factors, where it phosphorylates the transcription complex,

enabling it to go into productive elongation when the gene is required to be expressed. DRB prevents p-TEFb from phosphorylating the transcription complex, and consequently inhibits transcription. (30)

Another part of sequence level gene regulation is that genes are organized into "sections" of clusters on the genome of eukaryotes, and this begs the question of whether genes are regulated individually, or in cluster groups? Well-known clusters that have been extensively studied include the α-globin genes, β-globin genes, the *HOX* cluster and the histone gene battery. Spellman *et al* did a study on the expression profile in *Drosophila* under a wide variety of biological conditions and found that many functionally non-related proximal genes (20 to 200 kb of genes) display almost identical expression profiles. This implied that these genes might be co-regulated under similar experimental conditions. (38) Several other studies reported similar results. Increasing evidence is pointing to the fact that eukaryotic genome is not organized randomly, and that genes are clustered into groups. (29) How are these groups of genes co-regulated?

The first element that is responsible for switching clustered genes to an active state, is the cluster control regions such as the locus control region (LCR), which recruits histone modifying proteins to a certain region of the genome to induce a change in the chromatin structure. (29) LCRs have been found for a large number of loci.

The second element responsible for gene regulation in these clustered genes include the promotors and enhancers. (29)

The third element is the boundary element or insulator, which is responsible for the functional isolation of clustered genes from the rest of the genome. (29) This underscores that fact that genes are more organized in eukaryotes than previously thought.

1.3.2. Regulation at the level of chromatin

In the nucleus DNA is packaged in a dense structure called chromatin. The modification of the structure of chromatin is directly linked to gene expression. (40) This structural modification is achieved by reversible modification of the histones. (10) Examples of modifications include reversible acetylation of thirteen lysine residues in the N-terminal tail of the histone proteins, phosphorylation of three serine residues, and the methylation of three arginine residues in H3 and H4. Two residues can also be ubiquitinated in the C-terminal of H2A and H2B. (29)

Hyper-acetylation is related to the "open" configuration of the chromatin structure, and, accordingly, gene expression. (40) Methylation of Lys9, Lys27 and Lys35 of H3 leads to a "closed" configuration of the

chromatin structure, and subsequently repressed gene expression. This leads to what is called heterochromatin. (41) Methylation of Lys4 of H3 is associated with active gene expression. (42)

The class of enzymes that controls acetylation in the cell is the histone acetyltransferases (HAT) and histone deacetyltransferases (HDAC). (43) The methylation and demethylation is performed by a class of enzymes called histone methyltransferases. (44)

Other mechanisms that influence gene expression at the chromatin level is the chromatin remodelling enzymes. These enzymes change the local structure of chromatin by shifting nucleosomes, an energy-consuming process that requires ATP. Remodelling complexes can also influence nucleosome density by removing histones from nucleosomes. These remodelling complexes, which play a crucial role in nucleosome restructuring, were discovered in the 1990s, and can be divided into four categories based on their sequences: SWI/SNF, INO80/SWR1, ISWI and CHD. (45) These complexes do not target specific genes, but rather defined chromatin domains. The remodelling complex uses ATP hydrolysis to change nucleosome structure by either sliding the nucleosome along the DNA, or by completely removing it. (46) The SWI/SNF complexes slides the nucleosome on the DNA, irrespective of whether transcription should take place or not, and can also be described as a positive regulator of transcription. Positive transcriptional regulators induce transcription. Similar to SWI/SNF, Remodels Structure of Chromatin (RSC) complexes are positive regulator of transcription. Both these complexes contain bromodomains. Nucleosomes in the area of promotors, which are hyper-acetylated, are recognised and bound by these bromodomains, causing the recruitment of these remodellers to the acetylated lysine residues of the nucleosomes in promoters tagged for gene activation. (47)

The RSC complex is important in yeast cells transcription through the acetylation of nucleosomes surrounding promotors of genes transcribed by pol I, II and III, whereas SWI/SNF is only recruited to the acetylated nucleosomes of pol II genes. (47) The INO80/SWR1 remodellers contains an ATPase domain, which makes it distinct from other remodelling complexes. (47) The INO80 complex is involved in a wider role in both gene transcription and in DNA repair. This remodeller has no known histone recognition site. (47) The SWR1 complex swaps out the histone H2A for the H2A.Z isotype, and this change is known to open the chromatin, although it does not mean transcription will take place. (47) The ISWI remodellers are known as negative regulatory complexes and works in conjunction with other proteins (such as TUP1-SSN6) to repress gene expression. (47) This is by keeping nucleosomes in a deacetylated state. In a logical sense, these complexes lack a bromodomain, since they work against the effect of hyper-acetylated nucleosomes, so they probably work with certain regulators to locate its area of function. (47) The CHD remodellers contains a chromodomain that binds to methylated lysines of the nucleosome. A study was done in a mutant Chd1 remodeller yeast cell, and it was found that few genes were affected, which suggests it might work in conjunction with other remodelling complexes such as the SAGA complex. (47)

1.3.3. Regulation at the level of nuclear architecture and partition

The regulation of gene expression does not only transpire at the level of DNA and chromatin, but it also depends on its relationship to the different compartments and structures within the nucleus. Some locations within the nucleus promotes gene expression, and others are repressive to gene expression. (29) For example, when a gene is located in the periphery of the nucleus, it is repressed. (48)

This has to do with the arrangement and positioning of the chromosome within the nucleus, contact with the nuclear lamina, and the interaction of the chromatin structure with sub nuclear molecules and proteins. There are three different models proposed to describe the expression of non-orthologous genes. The first is the incidental model, the second the chromatin domain model, and the third is the three-dimensional space model. (49)

The incidental model specifies that if a transcription factor promotes a transcriptional response from a gene, that the adjacent genes, with no biological relationship to each other, will exhibit a similar transcriptional response. The chromatin domain model proposes that the opening of a chromatin domain or region facilitates the binding of transcription complexes to any promoter in the domain, and that the insulator complexes limit the propagation of this primed domain. Lastly, the three-dimensional space model proposes that, in order for a gene to be expressed, the chromosome region containing the gene must be moved into a specialised nuclear sub-compartment. Neighbouring genes will also be moved into this sub-compartment due to their contour proximity and will similarly be expressed. (50)

1.4. Physical gene clusters

In prokaryotes many genes are expressed from a single promotor as polycistronic mRNA, indicating that gene grouping is not random. Prokaryotes share an evolutionary ancestor with eukaryotes, which evolved into multicellular eukaryotic cells, which created the complex life forms that we know today. How do gene clusters in prokaryotes differ from those in eukaryotes? Clusters in prokaryotes are operons which consists of several genes whose expression are regulated by one promotor. Operons are transcribed into a single mRNA that encodes several genes, and the transcribed genes are often involved in different steps of a linked metabolic pathway. (51) For example, the lac operon in *Escherichia coli* is expressed to metabolize lactose through the expression of three genes (LacZ, LacY and LacA genes) from one promotor. (52)

There is significant evidence that gene location is also not random in eukaryotic organisms, and that an underlying clustering is, in large, present. In a study performed using the Kyoto Encyclopaedia of Genes and Genomes (KEGG), the inter-gene distance between genes involved in the same metabolic pathway were calculated and compared with the distance between the randomised distributions of the genes. It was shown

that in all species there were statistically significant gene clusters, and that although clusters of genes were not necessarily neighbouring genes, the constituent genes were relatively close in the genome. (53)

The definition of gene clusters varies. For example, in a study by Yi *et al,* a cluster was described as "a set of genes with the same common function that is closer to one another than by chance". (54) A cluster in this study is defined as a group of genes that have translation starting positions located within a minimum genomic distance of each other, and that exhibit a similar functional genomic property, such as a level of gene expression. (55) Is there any evidence for clusters in eukaryotic species?

There is significant evidence that gene position in eukaryotes is not random as mentioned previously, and that the genomes of eukaryotes are similarly divided into sections, defined as physical gene clusters. (38) This suggests that genes in eukaryotes are not randomly organized in the genome but are partitioned into co-regulated sections. (55) This does not imply that these genes are functionally connected, but it does imply that they might be co-regulated, and it is not only individual promotors, bound by transcription factors, that regulate individual genes. (38) There are many examples of such observations in eukaryotes, such as the silence mating type loci in *Saccharomyces cerevisiae*, the inactive X chromosome in mammals, and the β-globin gene locus in the erythrocyte cells of chickens. (56, 57, 58)

How is the terms co-expression and co-regulation defined? Co-expression refers to "a property that show similar spatial or temporal expression patterns". (53) Co-regulation refers to a set of genes that are regulated by the same biological mechanism. There is, however, a correlation between the expression (co-expression) and the regulation (co-regulation) of genes. Co-expressed genes can be expressed from any genomic location and does not imply that it is regulated through the same mechanism (co-regulation). Co-expression of adjacent genes is hypothesised to be regulated by the same mechanisms, which implies co-regulation. An example of regions that are regulated by similar mechanism is euchromatin or heterochromatin regions of the genome. (59)

Examples of physical gene clusters found in *S. cerevisiae* include two large clusters, the GAL and the DAL cluster, where the DAL cluster is the largest in the genome. The DAL cluster makes the yeast able to metabolise allantoin as a nitrogen source. This cluster contains six genes, which produces six proteins out of a total of eight proteins responsible for allantoin degradation. (3)

A study in *S. cerevisiae* by Cho *et al* they looked at the impact of cellular development on the expression of gene clusters. In that study they investigated the fluctuation in gene expression during the cell cycle of *Saccharomyces cerevisiae.* (61) The cell has to undergo multiple phases of "development" such as DNA replication, and the separation of the chromosomes during a cell cycle. When these processes are not properly regulated, genomic integrity is compromised, which is unfavourable to the cell. (62) The study showed that

approximately 25% of all the genes in the cell that are expressed during a similar cellular phase, are located adjacent to one another. (61)

In the worm *Caenorhabditis elegans,* 15% of its genes are located in operon-like clusters, which suggest that they share a similar regulatory mechanism. (63) In a study by Lercher, they compared microarray data from *C. elegans* and observed that the active genes were not randomly distributed in the genome, and that genes in the operons showed substantial levels of co-expression (63). Studies in *Arabidopsis thaliana* showed that gene clusters were associated with root and mitochondrial development. (64) A study by Bowles and Williams used microarrays to show that adjacent genes were often co-expressed. It was found that a significant number of genes involved in similar metabolic pathways, are located in proximity to each other. Of the 1891 genes that are allocated to various metabolic pathways in the *Arabidopsis* genome, 921 of them are within 10 genes from other genes in the same pathway. (65) In a study by Spellman and Rubin, microarray data for 88 biological conditions showed that 20% of *Drosophila* genes are found in clusters (10-30 genes). (38)

1.5. Evolutionary development of gene clusters in eukaryotes

There are evolutionary advantages to the development of physical gene clusters in a cell. These include the energy parsimony of remodelling chromatin in a single region as opposed to in many individual sites. The accessibility of transcription factors to open regions of chromatin can lead to the coherent expression of many essential genes that are involved in the same metabolic pathway. (53) Alternatively, these genes may simply be functionally related, co-ordinately expressed under a certain cellular condition, but not active in the same pathway.

The expression of these genes would have been energetically unfavourable if the gene positions were random. The non-random gene order makes the co-regulation of clustered genes possible for several different metabolic pathways. Yeast cells have the highest number of genes located in clusters that are involved in similar metabolic pathways, and this is mostly in regions with the lowest recombination rate. (53) The question is if genomic rearrangements occur with a similar frequency in more complex organisms? Evidence indicate there is a difference. The genomes of more advanced multicellular organisms are arranged more optimally. It has been proposed that evolution leans towards the development of co-regulated expression, and that this bias is an important reason for the evolution of gene order. (53) The gene order in the genome of a species has been proposed to be directed to maximise recombination frequencies of genes. Theoretical work has suggested that the recombination frequency between genes is influenced by their epistatic interaction. Epistasis refers to the phenomenon where genes are affected by one another, or that a mutation in one gene is influenced by a mutation in another gene, to result in a certain phenotype. (66)

There are more reasons why the formation of clusters is advantageous to species. If the genome was organized randomly, the expression of one gene might influence the expression of an undesired genes. Viruses might also be advantageous to non-random gene order, since insertion of viral DNA occurs in open regions of chromatin in the genome, and this might explain why the most highly expressed genes are located in genomic location with the highest gene density.

It is known that genes that are co-expressed are normally clustered in the genome of an organism, but the reason is not always clear. A study by Pal and Hurst looked at this from an evolutionary perspective. They divided the genome of *S. cerevisiae* into non-overlapping blocks of 10 genes. The gene expression variance of these blocks was determined for the essential genes in these blocks. If there was high variance, that would indicate high clustering. Expectedly the result indicate that the clustering of essential genes had high variance ($p<10^{-4}$) which suggested that essential genes are clustered in the genome, with the corollary that they are co-expressed. (67)

The study also speculated on why the genes were clustered. Firstly, it might be because of tandem duplication. Secondly, the evolutionary selection of genes of similar expression to be adjacent or in near proximity, and, lastly, to promote linkage disequilibrium. (67) The last reason suggests that there is a chance of low recombination between the genes in the cluster, while the first two reason suggests the opposite. The researchers then looked into the recombination rates of essential and non-essential genes in the blocks of 10 genes. They found that the essential genes had a lower rate of recombination, and that recombination is linked to the expression rate. (67) This does suggest that low recombination maintains gene order, but also suggests that recombination might lead to large-scale genomic rearrangement events and the introduction of structural mutations. This may be the evolutionary driving mechanism for the formation of these clusters. Although these findings are significant, there is not a significant amount of evidence relating to recombination rates in clusters in diploid organisms. (67)

Sugino investigated the effect of whole genome duplication in *S. cerevisiae* on genomic rearrangements, leading to the observed gene order. A whole genome duplication event (WGD) occurred in ancestral *Saccharomyces* some 100-200 million years ago. There is sequence data available before and after the WGD event. (68) It is speculated that the regulation of genes that are co-expressed drives the evolution of gene order through the nucleosome free regions (NFR), where RNA polymerase can initiate transcription. When two genes are adjacent after evolution of a specific gene order, it is the NFR in relation to the promotor of these genes that drives the fitness of this newly formed cluster. (68) It is found that genes that are co-expressed kept their position after the WGD event. (68)

Wong *et al* investigated the DAL cluster in yeast, which is the largest cluster and consist of six genes responsible for the allantion degradation. They looked at the formation of gene clusters that were not formed

through genome duplication, but through gene order rearrangements during evolution. (69) The investigators looked at genes that were homologous in closely related species, and found clusters to be conserved in the closely related *Saccharomyces sensu stricto* group. The arrangement of the genes in the cluster and the genomic location of the clusters, interestingly, differed. They further found that four of the DAL genes are single-copy genes that moved to the location in the cluster through gene duplication, and that the other two genes remained as progenitor genes at the same location in the cluster. This rearrangement of the genes in the cluster happened almost simultaneously. (69) They speculate that the DAL cluster formed to keep some genes in linkage disequilibrium, which can be described as an epistatic selection. Such areas normally have low recombination rates.

1.6. Project data

Functional genomics data is a gateway to observe physical gene clusters. With further downstream analysis of cluster expression in certain cellular conditions or stages, conclusions can be made about gene regulatory behaviour. Microarray and RNA-seq are examples of techniques that generate functional genomics data amenable to this approach. In this study microarray data was used.

Microarrays is an experimental technique where defined sequence nucleic acid primers are bound to individual spots in an array on a surface, which is used to identify and quantify the expression of a hybridized probe. (70) For this study, gene expression was measured at certain time points. One should firstly identify co-expressed genes that are in close proximity by selecting a cut-off expression fold change. (71) Secondly, one should statistically verify any co-regulatory relationships.

To look into the chromosomal interactions, experimental techniques such as Hi-C gives insight into the chromosomal configuration within the nucleus and show whether certain areas of chromosomes are in close spatial proximity. By overlapping these areas with differential gene expression data, insight may be gleamed into whether correlation between spatial proximity and gene activity may underlie any observed co-regulation.

The modification of the chromatin by acetylation causes the structure to open and enhance gene expression. One can identify these areas through the use of experimental procedures such as ChIP-seq, that maps the genomic location of acetylated histones directly. These locations can be superimposed on the differential gene expression data, to investigate any possible correlation and putative causality. Alternatively, ChIP-seq is also used to determine the genomic distribution of a specific protein. This allows an assessment of the possible causal role between the presence of the given protein and gene expression. Thus, this approach can indicate whether identified clusters may be co-regulated, using appropriate software and statistical tests.

1.7. Problem statement

Several studies have identified clustered gene expression in organisms using in-house scripts. There are not many studies that have developed a software tool for cluster detection based on a functional genomics property, such as gene expression level. Furthermore, not many studies have shown clustered gene expression over a time period as snap-shots of clusters of genes that are switched "on" or "off" under certain cellular conditions. Also, the question is how the regulatory mechanisms relate to chromatin structure associated with modifications such as acetylation, and what impact the chromatin configuration has on cluster regulation. In this study, I investigated the exit of *S. cerevisiae* from stationary phase and cell cycle re-entry following feeding of the cells with a glucose-rich medium. I specifically focus on whether clusters of genes are activated during this process of cell-cycle re-entry, and I finally ask whether these clusters are co-regulated, and if so, what mechanisms may possibly play a role?

1.8. Project justification

Studies indicated that most highly active genes form clusters, and that these clusters of genes have co-regulatory behaviour. Functional genomics data from *S. cerevisiae* in starvation conditions will be compared to the data obtained when the cell exists stationary phase and re-enters the cell-cycle, to observe whether clusters of genes are activated. To achieve this, a program, *Pyxis2*, will be developed, verified and compared to a similar program in the literature. This study will stimulate further future questions about what factors play a role in the regulation of the complex machinery involved with gene expression in eukaryotes.

1.9. Aims

The first aim of this study is to develop the tool *Pyxis2*, and compare it to another related tool, and to establish its utility in the use of real data. The tool will be developed in the programming language *Python.* Secondly, *Pyxis2* will be compared to *Cluster Locator,* which is also used in detecting statistically significant co-expressed clusters. Lastly the code will be applied to real data from *S. cerevisiae* as the cell exits stationary phase. The hypothesis is that clusters of genes are co-regulated due to epigenetic interactions and possible spatial genomic organisation as the cell exists stationary phase and re-enters the cell-cycle.

1.10. Objectives

1. Develop a genome "agnostic" software tool to analyse the statistical significance of clusters of genes with a similar quality. These include genes that are expressed, genes that are repressed, and so forth.

    a. Implement the hypergeometric distribution in the software code.

    b. Code appropriate classes to detect clusters from differential gene expression data.

c. Optimise the code to function on the *Linux* command-line.

2. Verify and characterize the software using synthetic data sets

   a. Compare with *Cluster Locator,* a similar program.

   b. Use a synthetic dataset to test the program

   c. Characterise the use of step size on the removal of false positive clusters.

3. Apply the software to study whether groups of genes are induced in a co-regulatory fashion when a yeast cell re-enters the cell-cycle

   a. Identify gene clusters in public data

   b. Perform a gene ontology and phylogenetic study of the detected clusters

   c. Retrieve Hi-C and ChIP-seq data generated from similar cell-cycle re-entry experiments from public repositories

   d. Identify any correlations between the Hi-C, acetylation ChIP-seq and the identified active clusters

CHAPTER 2

Development and verification of *Pyxis2*

2.1. Introduction

The aim is to develop a software tool that is genome agnostic that can be used to calculate the statistical significance of physical gene clusters in the functional genomics data of any organism. The gene clusters are defined by the experimental question and can be groups of genes that are transcriptionally up-regulated in response to the cellular environment, or genes that become transcriptionally repressed in response to, for instance, expression of a repressive heterochromatin protein. This tool will provide insight into the possible role of gene clusters and spatial proximity on gene co-regulation. *Pyxis2* will be coded in Python, a language that is widely used in the bioinformatics field today, which will facilitate adaptation and maintenance of *Pyxis2*.

In this chapter we will also compare *Pyxis2* to *Cluster Locator* (72) in terms of functionality and the type of statistical tests that are used by the programs.

2.2. Software design approach

2.2.1. Pyxis2

Gene expression analyses quantifies the expression of genes. By studying the level of expression of genes that are located within a local region in the genome, it is possible to identify clusters of genes that may be co-regulated. Note that groups of genes may respond to diverse transcriptional activators, and the upregulated expression of genes do not necessarily imply co-regulation. It may simply indicate that an experimental condition combines two possible activation sets that may well be separable in a different experimental condition. For instance, increase in temperature may induce both heat shock genes as well as general stress genes, whereas oxidative stress may induce only general stress response genes and not the heat shock genes, where the different conditions lead to the presence of different sets of transcriptional activators. Nevertheless, genes that exhibit cluster-like behaviour can be identified and the statistical significance of the cluster verified through appropriate tests. If the expression of adjacent genes is significantly different from genes in the wider environment under certain cellular condition, it might have the biological implication that this is a core response of the organism. This may indicate that the identified cluster(s) play a crucial role in maintaining cellular homeostasis for a specific cellular condition.

Pyxis2 is a program capable of identifying and statistically calculating co-expressed gene groups or clusters from any gene classification data, including gene expression data. The program reads a General Feature

Format 3 (GFF3) file, which has a strictly specified format, and a list of genes supplied by the used. (73) These lists of genes can be generated from gene expression data where all genes are up-regulated or down-regulated by a statistically significant degree, or may be lists of genes that display a higher degree of connectivity in spatial Hi-C data. (74) Any such list of genes corresponding to a defined experimental property can be used. In this thesis I refer to up-regulated genes as the genes in such a supplied list for reasons of clarity and consistency, although the gene lists can, of course, be based on any of a number of gene characteristics. *Pyxis2* makes no implicit assumption on the functional connectedness of the genes. It simply calculates the statistical significance of any physical clusters of genes that are identified in the supplied list of genes.

The GFF3 file contains information such as gene names, chromosome names, whether the feature is an introns or exon, and the start and end positions of the feature, of the annotated genome. Users have the option to remove false positive clusters using various false discovery tests.

False positives are erroneously categorised as a True result and is a general concern in the analysis of very large datasets. An example of such a test is the Benjamini-Hochberg (B-H) correction test. (75) Co-expressed gene clusters that are identified, are saved in Browser Extensible Data (BED) format and can be directly viewed in a genome browser such as *Integrated Genome Browsers* (IGB). (76, 77)

Pyxis2 is available at https://github.com/Louis-Conradie/Pyxis2_classes_2020.

2.2.2. Hypergeometric distribution

A number of studies have been published on the physical clustering of genes in a genome. (55) These studies generally made use of two approaches to calculate or estimate the statistical significance of a cluster. Raghupathy and colleagues made use of a binomial distribution, calculating the probability of a group of genes being in the cluster category. The applicability of the binomial distribution in this context is contentious, since it calculates the probability with replacement. Another approach was to simply generate random clusters of genes by computational means and estimate from the frequency of specific cluster sizes what the statistical significance of any proposed cluster is. Although this gives a better indication of the cluster significance, accuracy depends on the number of iterations, and it is often unclear how many such iterations researchers perform. (78)

We derived a statistical method from first principles, which transpired to be the hypergeometric distribution. The hypergeometric distribution is sampling from a finite population without replacement.

The hypergeometric distribution is given by the following equation:

$$P(X = x) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}} \quad 0 \leq x \leq n \leq N \qquad \text{Equation 1}$$

Where the parameters represent the binomial coefficients ${}_{M}C_{x}$, ${}_{N\text{-}M}C_{n\text{-}x}$ and ${}_{N}C_{n}$, where N is the population size, n is the number of draws, M is the number of successful states in the population, and x is the number of observed successes.

This can be applied to a linear chromosome where the genes are represented as holes in a beam.

Figure 2.1. Illustration of hypergeometric distribution represented as balls on a stick

If you have 5 balls (n=5), what is the chance of getting 4 balls (x=4) into the four holes (M=4) in the rectangle if there are 10 open holes (N=10)? The problem can be approached as follows: what are the total number of combinations in which 4 balls can be placed in four holes in the stippled rectangle? This is multiplied by the total number of combinations in which the remaining balls (5 – 4 = 1) can be placed in the holes outside of the stippled rectangle. This factor is divided by the total number of combinations possible using all the balls over the whole beam.

The number of combinations (${}_{n}C_{r}$) are calculated by $\frac{n!}{[(n-r)!r!]}$ and the over-all probability with

$$\frac{MCx \times N-MCn-x}{nCr} = \frac{\binom{4}{4}\binom{6}{1}}{\binom{10}{5}} = \frac{1\times6}{252} = 0.024$$

Thus the discrete probability of this event occurring is 0.024, and if we randomly observe a group of 4 consecutive genes being upregulated in a list of 5 upregulated genes on a chromosome that encodes 10 genes, then that cluster is statistically significant.

2.2.3 Program verification

The functionality of any scientific program requires careful and rigorous verification using synthetic datasets. The use of test data allows one to control the expected result, facilitating a comparison of the output of the program with the expected result. For this study a synthetic dataset from the human genome was generated to establish that *Pyxis2* can accurately process large genomes (>1000 genes).

2.2.4. Program benchmarking

It is also essential to compare the resource requirements, performance and accuracy of a program to any existing programs that can be used for a similar analysis. *Pyxis2* and *Cluster Locator* are programs that have similar functionality that utilize differential gene expression data to identify statistically significant gene clusters in the data. Both programs utilize similar computational methods to detect gene clusters from observed regulated genes but utilize two distinct statistical tests to validate the significance of identified clusters. *Cluster Locator* was developed by Obregòn *et al* and is available at http://clusterlocator.bnd.edu.uy/. (72) *Pyxis2* and *Cluster Locator* differs in terms of the manner in which they detect clusters and also the type of statistical tests the programs utilize.

2.2.5. Identifying potential gene clusters in Pyxis2

As a first step to calculate the statistical significance of gene clusters, potential clusters must be identified. There are a number of ways to approach this problem. One can randomly select groups of genes and calculate the statistical significance of each such potential cluster. However, such an approach may be computationally costly, particularly for extended lists of genes, and is limited by the choice of minimum and maximum cluster sizes, where clusters that lie outside of these limits are missed. An alternative is a hierarchical clustering approach, where single gene clusters are progressively merged based on a maximum distance rule. In this latter approach the contour distance between upregulated genes are determined, and genes that are within a specified maximum distance are placed in the same potential cluster. This distance is specified as a step size, which is the maximal gene-to-gene distance of up-regulated genes, ignoring gene length. The step size directly influences the number of clusters that are observed and also the distribution of the statistical significance of clusters for a certain number of genes. Optionally, false discovery tests can be applied to the results.

2.3. Method

2.3.1. Software development

The program *Pyxis2* was developed using *Python* version 3.6.1 (*Python Software Foundation, Beaverton,USA)* on the *Ubuntu Linux* operating system version 16.04 (*Canonical Group Limited, London, UK*). *Pyxis2* is based on the original *Pyxis,* which was developed by Patterton and colleagues. (55) *Pyxis2* was coded in Python and, unlike *Pyxis*, is genome agnostic, and can be applied to data for any species for which a GFF3 format genome annotation file is available. *Pyxis2* is open-source and launches from the *Ubuntu* command-line. The code was developed on a *DELL* computer (Round Rock Texas, US), with 16 GB of RAM memory and an *Intel Xeon* CPU (E5-2630 v3 @ 2.40GHz × 16) (Santa Clara, California, USA).

Where input or output file content, code or command-line entries are shown, it will be shown in Courier New font text.

2.3.2. Data analysis

Pyxis2 was used to analyse synthetic dataset on the *Ubuntu* operating system version 16.04 (*Canonical Group Limited, London, UK*) via the *Linux* command-line interface (CLI). The *Integrated Genome Browser* (77) was used to visually represent the obtained results.

2.3.3. Construction of figures

Figures were generated using Python version 3.6.1 and *matplotlib* (*Python Software Foundation, Beaverton, USA*).

2.3.4. Workplan

The following approach was followed in developing and verification of the program:

1. Code the program in *Python.*
2. Verify the program with synthetic datasets
3. Benchmark the program with similar programs.
4. Characterise the impact of selectable parameters, such as gene step size in identifying clusters, on the output of the program.

2.4. Development of *Pyxis2*

2.4.1. Work-flow of *Pyxis2*

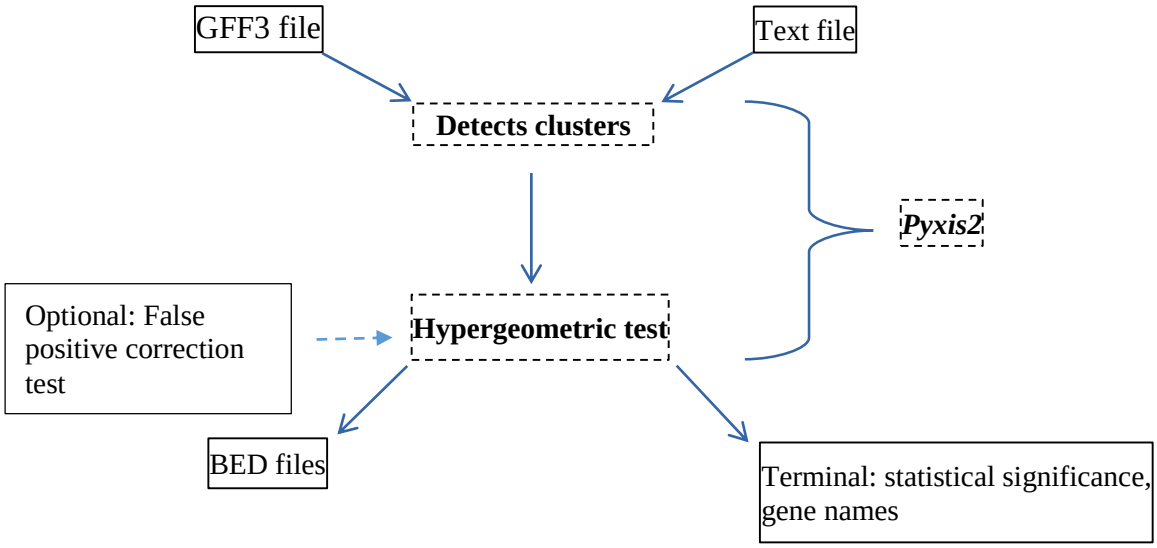


Figure 2.2. Workflow of Pyxis2. Detects clusters and then performs hypergeometric test. There is also an option of doing a false positive correction test to remove any false positives

As illustrated in Figure 6, *Pyxis2* uses an input GFF3 file and a text file with a list of genes that must match the gene identifiers in the GFF3 file. Output data for *Pyxis2* are two BED format files that contain the chromosome names, respective gene regions, and a colour scheme representative of the statistical significance of every detected cluster.

The first BED file contains the cluster regions, and second BED file contains the names of the genes in the clusters. The viewer can thus observe the start and end position of clusters as well as the regulated genes within each cluster independently.

An important thing to note is that when a false discovery test is implemented, the BED files will contain the adjusted p-value. *Pyxis2* uses standard file formats compatible with most workflows/pipelines.

2.4.2. Input files

2.4.2.1. List of regulated genes

This text file is the list of regulated genes and is composed of lines where each line contains a single gene name terminated with a LF character (CRLF on Windows). An example of such a list of genes from *S. cerevisiae* is:

YAL001C
YAL002W
YAL003W
YAL004W
YAL005C
YAL007C
YAL008W
YAL009W

2.4.2.2. GFF3 file

Pyxis2 requires a GFF3 format file of the annotated genome to be analysed for gene clusters. An example file from *S. cerevisiae* is given below.

```
##gff-version 3
##sequence-region   I 1 230218
|…
##sequence-region   XV 1 1091291
##sequence-region   XVI 1 948066
#!genome-build SGD R64-1-1
#!genome-version R64-1-1
```

```
#!genome-date 2011-09
#!genome-build-accession GCA_000146045.2
#!genebuild-last-updated 2011-12
I     SGD   chromosome 1     230218     .     .     .
      ID=chromosome:I;Alias=BK006935.2
###
I     SGD   gene  335   649   .     +     .
      ID=gene:YAL069W;biotype=protein_coding;description=Dubious
open reading frame%3B unlikely to encode a functional protein%2C
based on available experimental and comparative sequence data
[Source:SGD%3BAcc:S000002143];gene_id=YAL069W;logic_name=sgd
I     SGD   mRNA  335   649   .     +     .
      ID=transcript:YAL069W;Parent=gene:YAL069W;biotype=protein_codi
ng;transcript_id=YAL069W
I     SGD   exon  335   649   .     +     .     Par-
ent=transcript:YAL069W;Name=YAL069W.1;constitutive=1;ensembl_end_ph
ase=0;ensembl_phase=0;exon_id=YAL069W.1;rank=1
I     SGD   CDS   335   649   .     +     0
      ID=CDS:YAL069W;Parent=transcript:YAL069W;protein_id=YAL069W
###
I     SGD   gene  538   792   .     +     .     ID=gene:YAL068W-
A;biotype=protein_coding;description=Dubious open reading frame%3B
unlikely to encode a functional protein%2C based on available
experimental and comparative sequence data%3B identified by gene-
trapping%2C microarray-based expression analysis%2C and genome-wide
homology searching [Source:SGD%3BAcc:S000028594];gene_id=YAL068W-
A;logic_name=sgd
I     SGD   mRNA  538   792   .     +     .     ID=transcript:YAL068W-
A;Parent=gene:YAL068W-
A;biotype=protein_coding;transcript_id=YAL068W-A
```

2.4.3. Output files

2.4.3.1 BED file

The output BED files are formatted according to the specification. (76) An example output BED file is given below. It lists the chromosome name of every cluster that was detected, as well as the start and end positions of the cluster, statistical significance and also the 8-bit RGB colour values corresponding to the reported p-value. Only the red RGB component is used. The more significant the p-value, the higher the R channel value.

```
track name= up0_clusters description="Item RGB demonstration"
itemRgb="On"
I     31567      192256     cluster    1.92E-02   0     0     0
      190,0,0
II    247010     360187     cluster    3.00E-02   0     0     0
      130,0,0
II    450881     541021     cluster    1.73E-03   0     0     0
      255,0,0
II    553543     578984     cluster    4.65E-02   0     0     0
      80,0,0
II    738582     804475     cluster    2.80E-02   0     0     0
      140,0,0
III   6479       71367      cluster    2.22E-02   0     0     0
      170,0,0
III   102075     138996     cluster    2.77E-02   0     0     0
      140,0,0
IV    282112     376480     cluster    1.13E-02   0     0     0
      250,0,0
IV    678241     825776     cluster    6.45E-03   0     0     0
      255,0,0
IV    843569     1015702    cluster    1.19E-02   0     0     0
      250,0,0
IV    1279210    1428120    cluster    3.74E-02   0     0     0
      110,0,0
IX    16784      83041      cluster    3.03E-03   0     0     0
      255,0,0
IX    93619      144085     cluster    1.63E-02   0     0     0
      210,0,0
IX    157385     183127     cluster    1.70E-02   0     0     0
      210,0,0
```

2.4.3.2. Pyxis output

The terminal screen output after executing Pyxis2 is shown in Fig. 2.3 below:

```
The search string for the GFF file is:  ID=gene:
The database string for the GFF file is:  Name=
The cut-off p-value is:  0.01
The minimum window for the detection of gene clusters is:  5
_____________________________________________________________________________________
chromosome: 1
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355,
1356, 1357, 1358]
number balls: 24
total length: 2098
number of clusters: 2
=============================================================+=======================================
chromosome length: 2098
total genes: 24
clusters: [[0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353,
1354, 1355, 1356, 1357, 1358]]
gene cluster: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
window size: 10
genes in window: 10
gene    names:    ['ENSG00000186092',    'ENSG00000284733',    'ENSG00000284662',    'ENSG00000187634',
'ENSG00000188976',   'ENSG00000187961',   'ENSG00000187583',   'ENSG00000187642',   'ENSG00000188290',
'ENSG00000187608']
significance: 4.40E-21
gene cluster: [1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358]
window size: 14
genes in window: 14
gene    names:    ['ENSG00000169231',    'ENSG00000173171',    'ENSG00000177628',    'ENSG00000160767',
'ENSG00000116521',   'ENSG00000176444',   'ENSG00000143630',   'ENSG00000143627',   'ENSG00000160752',
'ENSG00000160753', 'ENSG00000116539', 'ENSG00000125459', 'ENSG00000163374', 'ENSG00000132676']
significance: 5.58E-30
=============================================================================================
```

Figure 2.3. Terminal output for Pyxis2

After execution, Pyxis2 displays the settings used in the analysis. The regulated genes in the genome and also the positions of the clusters are shown, with the significance of the clusters. The numerical values (0,1,2,3) represent the "positions" of each gene in relation to each other regardless of its length or the level of expression.

2.5. Modules and Command-line interface of *Pyxis2*

2.5.1. Argparse

The module *argparse* allows for the addition of command line arguments and parsing. Command-line interfaces which serves as interface for the input makes it possible for the users to easily change default settings in the code. The a*rgparse* module is initialized by creating the *ArgumentParser* object.

```
# In[28]:
parser = argparse.ArgumentParser(description='Pyxis detects
clusters of regulated genes and returns the confidence')
parser.add_argument('-v', metavar='v', type=str, help="GFF file")
#GFF file
parser.add_argument('-s', metavar='s', type=str, help="List of
regulated genes") #file with upregulated genes
parser.add_argument('-w', metavar='w', type=str, nargs='?',
default='None', const='None', help='Statistical test used to detect
clusters')
```

```
parser.add_argument('-i', metavar='i', type=str, nargs='?',
default='ID=gene:', const='ID=gene:', help='Search string used to
identify Genes in GFF file (default: ID=gene:)')
parser.add_argument('-k', metavar='k', type=str, nargs='?',
default='Name=', const='Name=', help='Search string used to
identify Gene name in GFF file (default: Name=)')
parser.add_argument('-p', metavar='p', type=float, nargs='?',
default=0.01, const=0.01, help='Lower P-value used to make
statistical test more rigorous (default: 0.01)')
parser.add_argument('-q', metavar='q', type=float,
nargs='?',default=0.05, const=0.05, help='Maximum p-value to make
statistical test more rigorous (default:0.05)')
parser.add_argument('-l', metavar='l', type=int, nargs='?',
default=5, const=5, help='Minimum window to detect gene clusters
(default: 5)')
args = parser.parse_args()
print("The search string for the GFF file is: ", args.i)
print("The database string for the GFF file is: ", args.k)
print("The cut-off p-value is: ", args.p)
print("The minimum window for the detection of gene clusters is: ",
args.l)
GFF_file = args.v
files = args.s
idstring = args.i
sig_value = args.p
minimum_window = args.l
```

The first parameter in the method is created to implement flags in the command-line and are defined as boolean operators. This also makes the interpretation of the command-line easier. The *metavar* parameter in the method provides a different name for optional arguments in the *help* message and secondly the *type* argument indicates the type of input parameter the command-line requires. Lastly the *help* parameter in the method serves as navigation when any of the input files are faulty or when the command-line is not implemented correctly. The first two arguments both require input files and are mandatory. The first parameter requires the GFF3 file for the particular species and the second parameter requires the text file containing the list of gene names. The last three parameters are optional, and if no value or strings are given the default parameters will be implemented.

The third parameter is a default string specified for a type of GFF3 file implemented and is also the first optional parameter in the command-line. The string "*ID=gene*" can be changed to, for example, "*ID=SGD*", as may be required for GFF3 files representing annotations from different genomes. The other parameter *k* contains the default string "*Name=*" which is always present in lines for genes which are protein coding. If the user opts to remove string from the parameter dubious ORFs and pseudo-genes can also be retrieved.

The fifth parameter in the command-line is used to set maximal significance parameter as cut-off p-value for any significant cluster. The sixth parameter is the value which will indicate the most significant clusters from the least significant. Lastly, the seventh parameter is the step size at which the code detects clusters.

These parameters serve as input to the *Pyxis2* program. The *parse_args*() method inspects the command-line and convert each argument to the appropriate type and the command-line arguments are then set to each of the various parameters in the program.

The command-line arguments which Pyxis2 uses after the import of the *Argparse* module is:

```
Requirements:
      Python 3.9.0
      numpy
      statsmodels
      matplotlib

Description :
      Pyxis2 is a command line utility use for calculating something:

Usages :
      Pyxis2.py -v [GFF File] -s -w -i -k -p -q -l
      -v         GFF3 File
      -s         Text File Containing Regulated Genes
      -l         Step Size, Amount of Genes Apart Equaling a Cluster
      -k         Open Reading Frames Type
      -i         String to Search For
      -w         False Discovery Rate
      -p         Lower p-value, most significant value
      -q         Top p-value, Cut off Value
Example :
      Pyxis2.py -v Example.gff3 -s RegulatedGenes.txt -l 2 -k 1 -i
SearchString -w 2 -p 0 -q 10
```

```
Pyxis2.py -v Example.gff3 -s RegulatedGenes.txt -l 2 -k "" -i
SearchString -w 2 -p 0 -q 10
```

Next, we will be discussing the different classes that Pyxis2 imports to perform several functions.

2.5.2. Classes

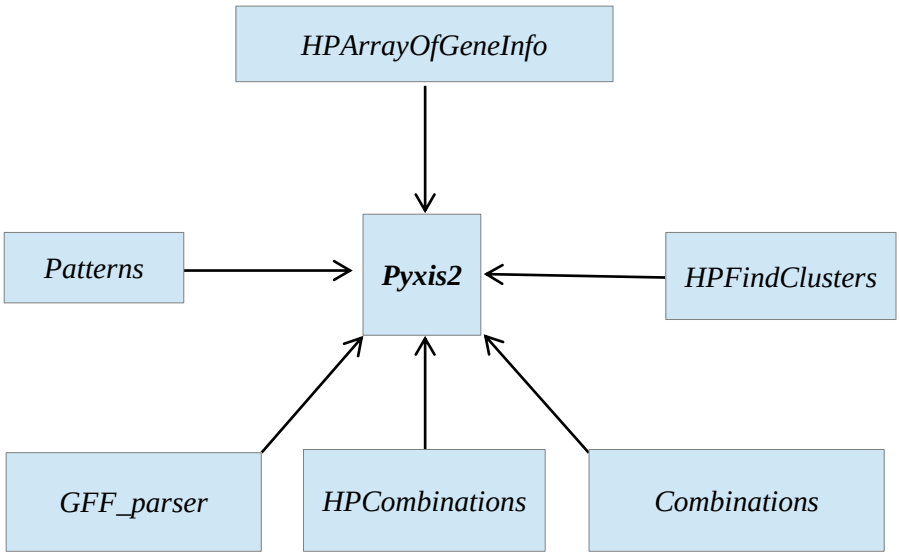


Figure 2.4. Different classes of Pyxis2

Pyxis2 makes use of 6 classes. The first class is *GFF_parser* that contain functions to read the GFF3 file, and extract the relevant information such as chromosome name, gene names, as well its corresponding start and end positions and the chromosome length. The *Patterns* class reads the list of regulated genes. Chromosomes are then represented as a binary string of length equal to the number of genes in the chromosome. Genes that were present in the read list of genes are tagged by flipping the bit at the corresponding position to 1. The distribution of 1s is used to identify clusters.

The class *HPFindClusters* encodes functions that identify potential clusters using the step size selected by the user. The class *HPArrayOfGeneInfo* provides functions to find the names of the genes in identified putative clusters. The class *HPCombinations* implements a hypergeometric distribution to calculate the statistical significance of each cluster. The class *Combinations* is similar to *HPCombinations*, but is only applied to large genomes (>1000 genes), using wider data types.

2.6. Results

2.6.1. Verifying the hypergeometric distribution of *Pyxis2*

Pyxis2 makes use of a hypergeometric distribution to calculate the statistical significance of putative physical gene clusters. To confirm the correct calculation, the clusters were generated from the *S. cerevisiae* genome, and the statistical significance of the clusters computed by *Pyxis2* and compared to the manually computed values. For example, a putative cluster of 5 genes were generated with 2 up-regulated genes. The gene list text file contained the entries:

YOR303W
YOR306C

The location of these genes is at positions 397 and 401 of a total of 472 genes on a chromosome.

The sample size for the population is 472 (*N*). This is the number of protein coding genes on the chromosomes. Thus, using equation 1, above, N = 472, n = 2, M = 5 and x = 2. The probability mass function of the hypergeometric distribution is thus given by:

$$\frac{\binom{5}{2}\times\binom{472-5}{2-2}}{\binom{472}{2}} = \frac{10\times 1}{111156} = 8.99\times 10^{-5}$$

Thus, the probability *p* that the 2 up-regulated genes on a chromosome with 472 genes appear within a block of 5 genes is 8.99×10^{-5}. This corresponds to the statistical significance calculated in *Pyxis2* for the detected cluster.

The program also needs to be compared to similar programs to indicate if the program is an improvement on functionality. *Cluster Locator* is a similar program to Pyxis2 and will be used to in the next part of the study to compare both programs.

2.6.2. Comparison with *Cluster Locator*

2.6.2.1. Workflow of *Cluster Locator*

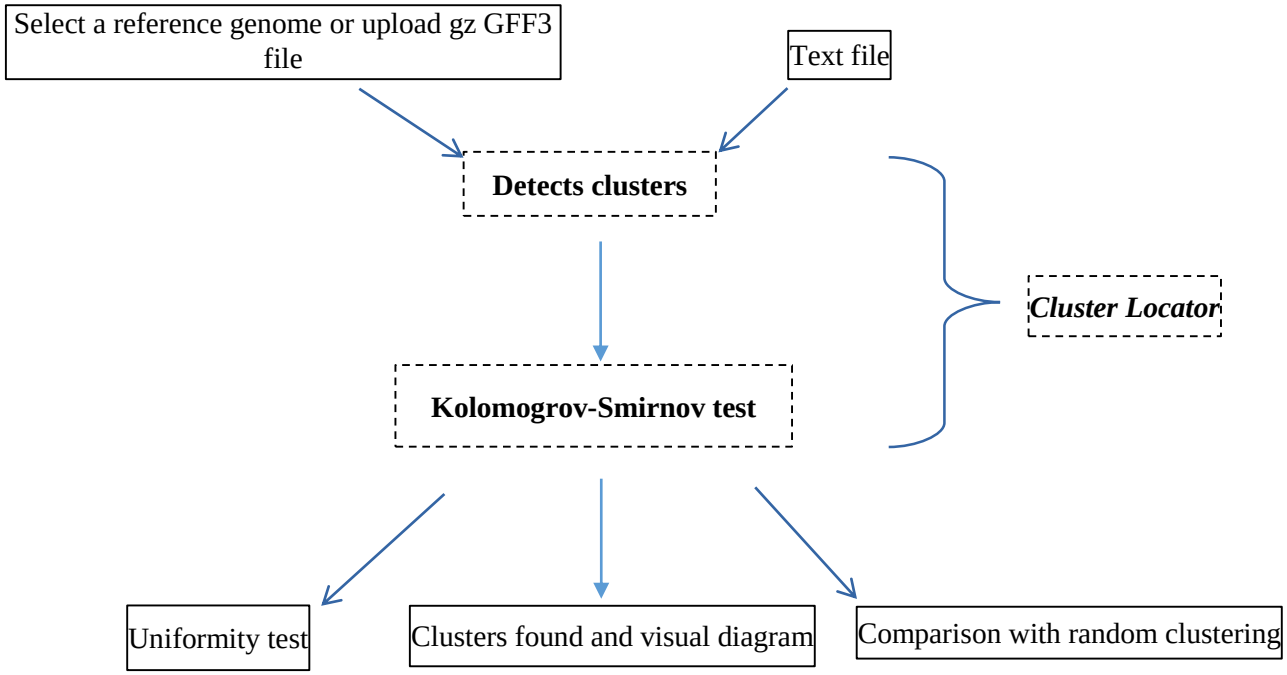


Figure 2.5. The Workflow of Cluster Locator. The program first identifies putative clusters and then performs Kolomogrov-Smirnov to estimate the statistical significance of the clusters.

Cluster Locator, published by Obregon and colleagues in 2018 (72), does not make use of the exact numeric calculation of cluster probabilities based on discrete values. *Cluster Locator* generates 1000 random distributions of the same number of query genes. The random distributions are assumed to be uniform, but this assumption is not tested. The distribution of the positions of sections of query genes are then subjected to a 2 sample Kolmogorov-Smirnov (K-S) test against the assumed uniform distributions, and the fit to uniformity expressed as the K-S D-statistic and p-value. The publication or supplementary material does not specify how these sections are defined. Use of the tool suggests that the sections are not equivalent to putative clusters, but whole chromosome arms. (72)

Next, putative clusters are identified in the 1000 random distributions of a number of genes equal to that of the query list. It is then verified that the percentage of genes that lie within putative clusters in the generated 1000 samples conform to a normal distribution. The mean and standard deviation of this normal distribution is calculated. The statistical significance of the percentage of clustered genes observed in the query list is then calculated as a *p*-value from the parameters of the normal distribution. Note that *Cluster Locator* simply provides a single significance value for the entire cluster dataset, and not on a per cluster basis. The 2004

paper by Chang *et al* that introduced the use of the hypergeometric distribution to calculate the discreet statistical significance of physical gene clusters, is not cited. (55)

2.6.2.2. Code availability

There is no open-source code available for *Cluster Locator*, but a web-based application is freely available at http://clusterlocator.bnd.edu.uy/ .*Pyxis2* is open source and available at the following link: https://github.com/Louis-Conradie/Pyxis2_classes_2020/blob/main/.

2.6.2.3. Synthetic dataset to compare both programs

A synthetic dataset generated from a *Homo sapiens* GFF3 file was used to compare the two programs.

The query list consisted of the first ten genes from chromosome 1, and a few genes from chromosome ten. A step size of 5 was implemented for both datasets.

The following genes were used:

```
['ENSG00000186092','ENSG00000284733','ENSG00000284662','ENSG0000018
7634','ENSG00000188976',
'ENSG00000187961','ENSG00000187583','ENSG00000187642','ENSG00000188
290',
'ENSG00000187608','ENSG00000180525','ENSG00000107929','ENSG00000107
937','ENSG00000148377',
'ENSG00000067064','ENSG00000047056','ENSG00000185736','ENSG00000067
057','ENSG00000107959', 'ENSG00000067082',
'ENSG00000165568','ENSG00000187134']
```

Terminal output of *Pyxis2*:

```
The search string for the GFF file is:  ID=gene:
The database string for the GFF file is:  Name=
The cut-off p-value is:  0.01
The minimum window for the detection of gene clusters is:  5
_____________________________________________________________________
_____________________________
chromosome: 1
```

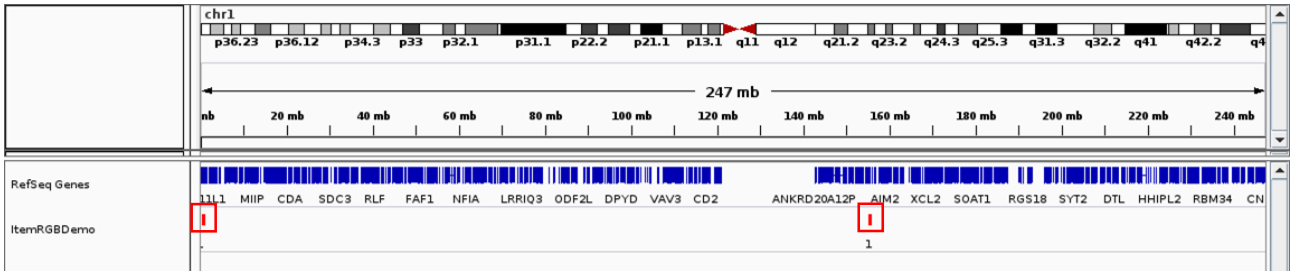
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 1345, 1346, 1347, 1348, 1349, 1350,
1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358]
number query genes: 24
total length: 2098
number of clusters: 2
================================================================
============================
chromosome length: 2098
total genes: 24
clusters: [[0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [1345, 1346, 1347, 1348,
1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358]]
gene cluster: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
window size: 10
genes in window: 10
gene names: ['ENSG00000186092', 'ENSG00000284733',
'ENSG00000284662', 'ENSG00000187634', 'ENSG00000188976',
'ENSG00000187961', 'ENSG00000187583', 'ENSG00000187642',
'ENSG00000188290', 'ENSG00000187608']
significance: 4.40E-21
gene cluster: [1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352,
1353, 1354, 1355, 1356, 1357, 1358]
window size: 14
genes in window: 14
gene names: ['ENSG00000169231', 'ENSG00000173171',
'ENSG00000177628', 'ENSG00000160767', 'ENSG00000116521',
'ENSG00000176444', 'ENSG00000143630', 'ENSG00000143627',
'ENSG00000160752', 'ENSG00000160753', 'ENSG00000116539',
'ENSG00000125459', 'ENSG00000163374', 'ENSG00000132676']
significance: 5.58E-30
================================================================
============================

________________________________________________________________
_______________________________
chromosome: 10
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
number query genes: 12
total length: 762
```

```
number of clusters: 1
================================================================================
============================
chromosome length: 762
total genes: 12
clusters: [[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]]
gene cluster: [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
window size: 12
genes in window: 12
gene names: ['ENSG00000180525', 'ENSG00000107929',
'ENSG00000107937', 'ENSG00000148377', 'ENSG00000067064',
'ENSG00000047056', 'ENSG00000185736', 'ENSG00000067057',
'ENSG00000107959', 'ENSG00000067082', 'ENSG00000165568',
'ENSG00000187134']
significance: 1.36E-26
================================================================================
============================

________________________________________________________________________________
_____________________________
.
.
.
chromosome: Y
[]
number query genes: 0
total length: 45
number of clusters: 0
```

Visual output for the BED files on Genome Browser as seen in figure 2.6.



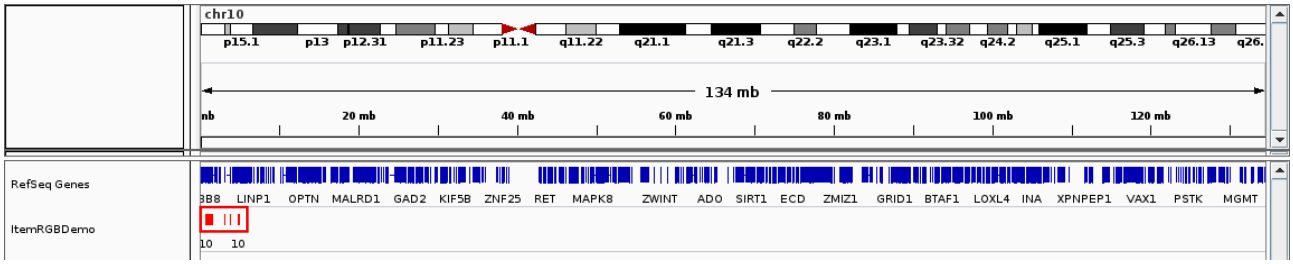


Figure 2.6. Changes in number of cluster detected against different step sizes

Visual output of *Cluster Locator:*

Table 2.1: Detected clusters using *Cluster Locator*

Gene_id	Chromosome	Starting position	Cluster_id
ENSG00000187961	1q	960587	1
ENSG00000187583	1q	966497	1
ENSG00000187642	1q	975204	1
ENSG00000148377	10p	1018907	2
ENSG00000067064	10p	1039908	2
ENSG00000047056	10p	1049538	2
ENSG00000185736	10p	1177318	2
ENSG00000067057	10p	3066333	3
ENSG00000107959	10p	3137728	3
ENSG00000165568	10p	4786629	4
ENSG00000187134	10p	4963253	4
ENSG00000107929	10p	806914	5
ENSG00000107937	10p	988019	5

Table 2.2: Statistical results using the K-S test

Chromosomal_segment	D	p_value
10p	3.1635e-01	1.4460e-01
1q	7.2668e-01	8.0656e-05

Table 2.3: Mean, standard deviation and p-value for the detected clusters

Mean	Std	p_value
1.38	3.7828441561753388	< 1.00e-10,65

Pyxis2 has two output files: the terminal results and the BED files. The following parameters were also identified: the total number of genes, individual clusters and their corresponding span, and the query genes present in the clusters.

Firstly, the results show that 3 clusters were detected. Two clusters on chromosome 1 and one cluster on chromosome 10. The *p*-value calculated for each cluster indicate that the clusters are statistically significant. This is indicated in figure 2.6 by the red bands. The Bonferonni test was applied to the list of identified clusters, and is also displayed by *Pyxis2*.

Cluster Locator has three different outputs: the clusters found, the uniformity test and a statistic analysis of the results. There were five different identified clusters for chromosome one and ten. Note that although 4 clusters are identified on chromosome 10, the test for uniformity include all 4 clusters simultaneously, and provides a single D_{max} and *p*-value for the section as a whole. The likelihood that 65% of a random list of 22 human genes would be in one or more clusters is deduced from the parameters of a normal distribution constructed from 1000 random selections of 22 genes and calculated at $p<1\times10^{-10}$. Note that this statistic gives the significance of the size of the total clustered population and provides no insight into the statistical significance of individual clusters, information that *Pyxis2* does provide.

A crucial feature of both *Pyxis2* and *Cluster Locator* is the use of a step size or maximum gap when identifying putative clusters in sets of genes. It is obvious that using larger step values is less stringent, i.e., a larger number of clusters are likely to be identified. I therefore investigated the impact that step size (max-gap) has on the identification of clusters in Pyxis2 and in *Cluster Locator*.

2.6.3. Effect of step size on cluster detection

The effect of step size on cluster detection was implemented through generating increasing numbers of random genes for each respective step size 1 to 5. Random genes were generated across the whole genome in the following order: 25, 50, 100, 200, 250 and 300 out of 1000 random genes (N) and pulled into *Pyxis2*.

Table 2.4: The effect of the different step size on the number of clusters detected

Step size	5	4	3	2	1
25 genes	1	1	1	1	0
50 genes	2	2	1	0	0
100 genes	8	7	6	4	1
200 genes	19	19	18	16	7
250 genes	27	26	23	21	15
300 genes	25	24	21	19	13

Table 2.4 indicates that, as expected, an increase in the number of random genes increase the number of clusters, up to a peak value. These clusters may include false positives. The decrease in the number of clusters at the highest value of genes is likely due to the merging of clusters allowed by the extra genes present in the distribution.

I next investigated the impact of the correction for false positives with the Benjamini-Hochberg (B-H) test on the relationship between step size and the number of identified clusters. The B-H test is used to remove any false positive results that is increasingly likely with the generation or analysis of very large data sets. When very many samples in a normal distribution are observed, it becomes increasingly likely to observe a sample that lies outside 3σ. Random genes were pulled into *Pyxis2*, and filtered with a B-H test after identification of clusters.

Table 2.5: The effect of the BH FDR test on the number of detected clusters

Step size	**5**	**4**	**3**	**2**	**1**
25 genes	1	1	1	1	0
50 genes	2	2	1	0	0
100 genes	8	7	6	4	1
200 genes	8	19	18	16	7
250 genes	16	12	15	20	15
300 genes	0	4	4	11	13

As expected, the number of clusters decreased significantly for each respective step size, particularly for larger gene populations. This result is visually represented in Fig 2.7.

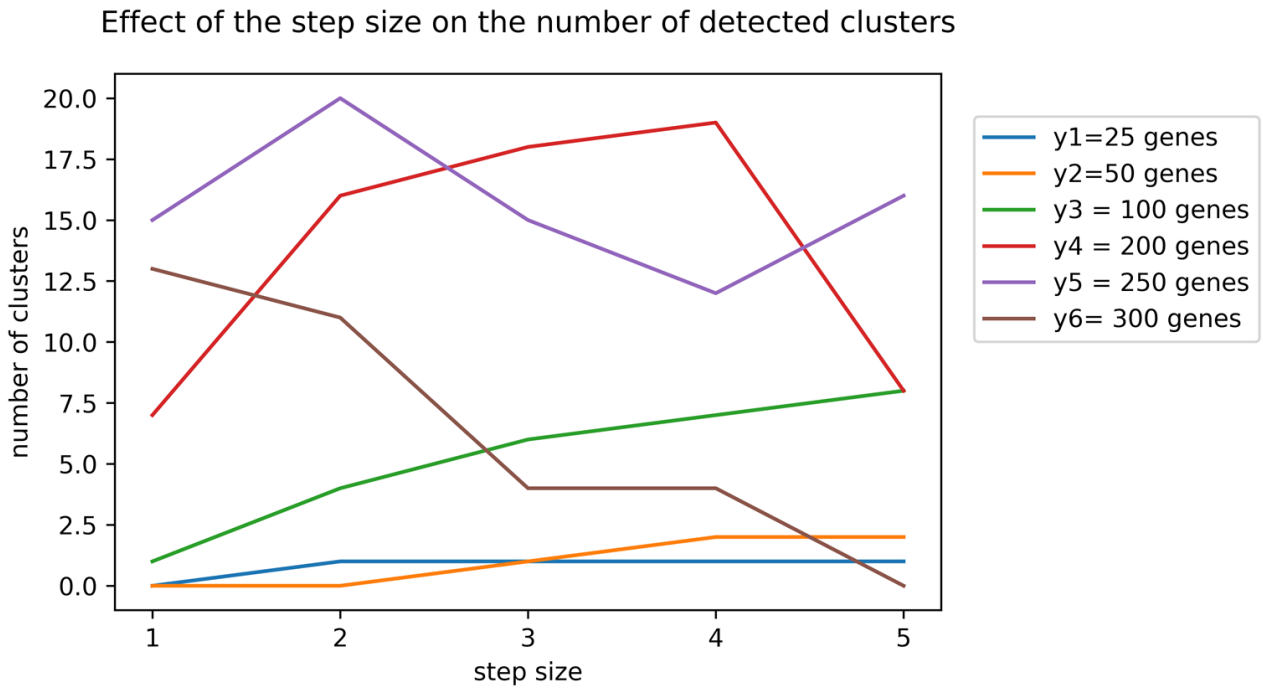


Figure 2.7. Changes in number of clusters detected against different step sizes

Few clusters were detected at low gene count (25 and 50) for all the respective step sizes because the data is random and is less chance that random clusters can be generated. At 100 genes the number of clusters increased

as the step size increases. Both at 200 and 250 genes there was optimal step sizes where the number of clusters detected reached a plateau and decreases again. At 300 random genes the number of false positive clusters is at a maximum.

I next looked at the distribution of *p*-values for clusters identified with different step sized in different sized sets of genes. The result is shown in Fig 2.8-2.12.

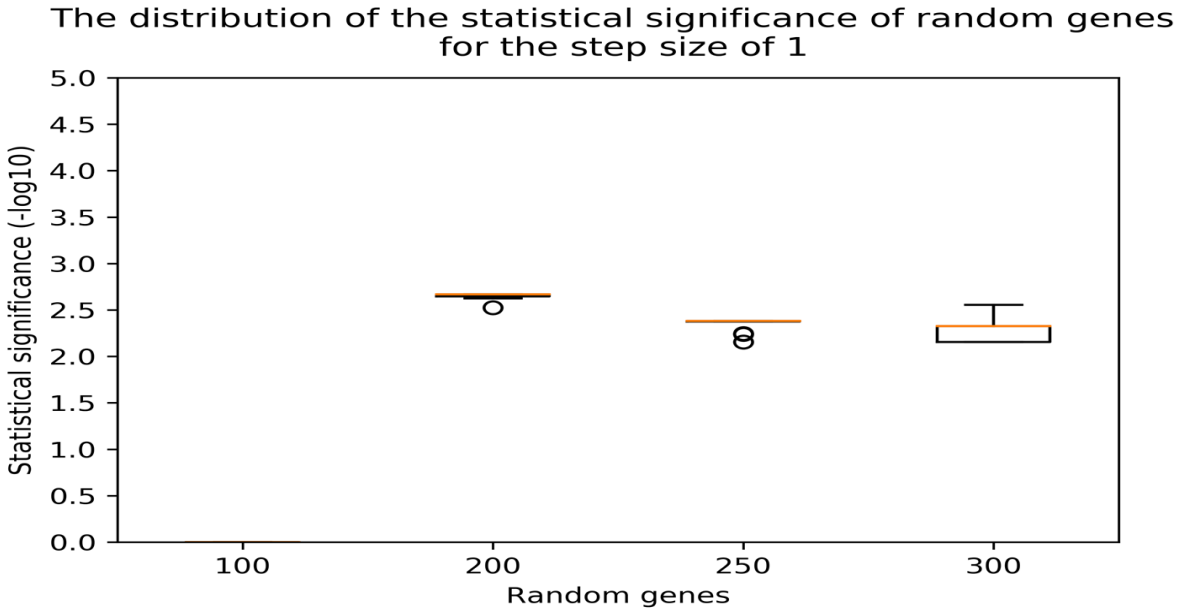


Figure 2.8. Distribution of statistical significance for random genes at step size of 1

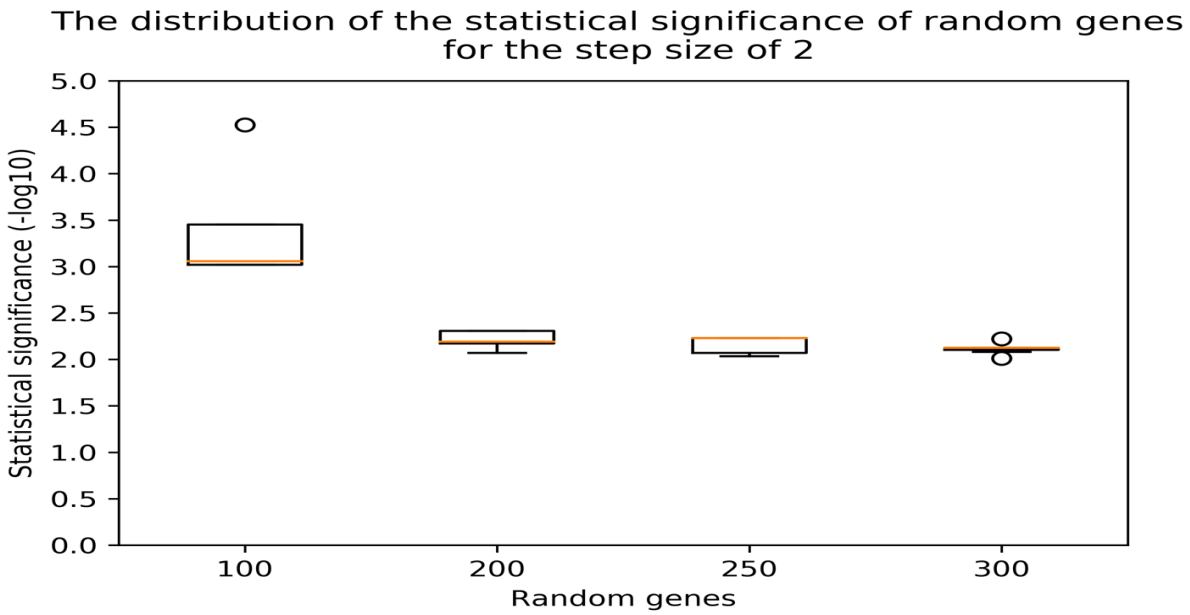


Figure 2.9. Distribution of statistical significance for random genes at step size of 2

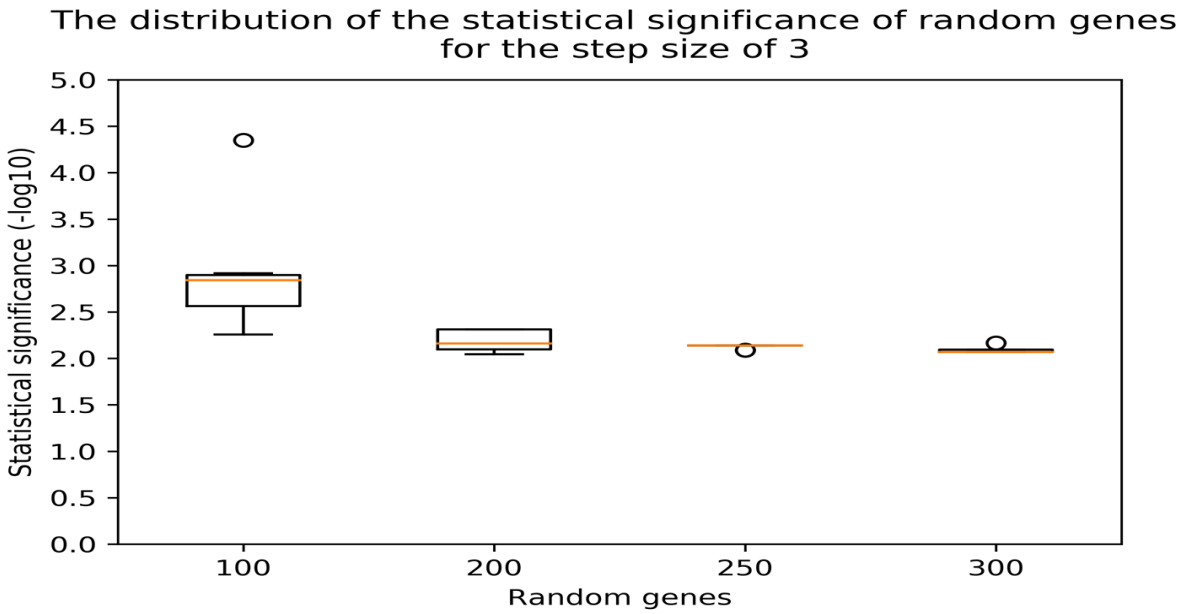


Figure 2.10. Distribution of statistical significance for random genes at step size of 3

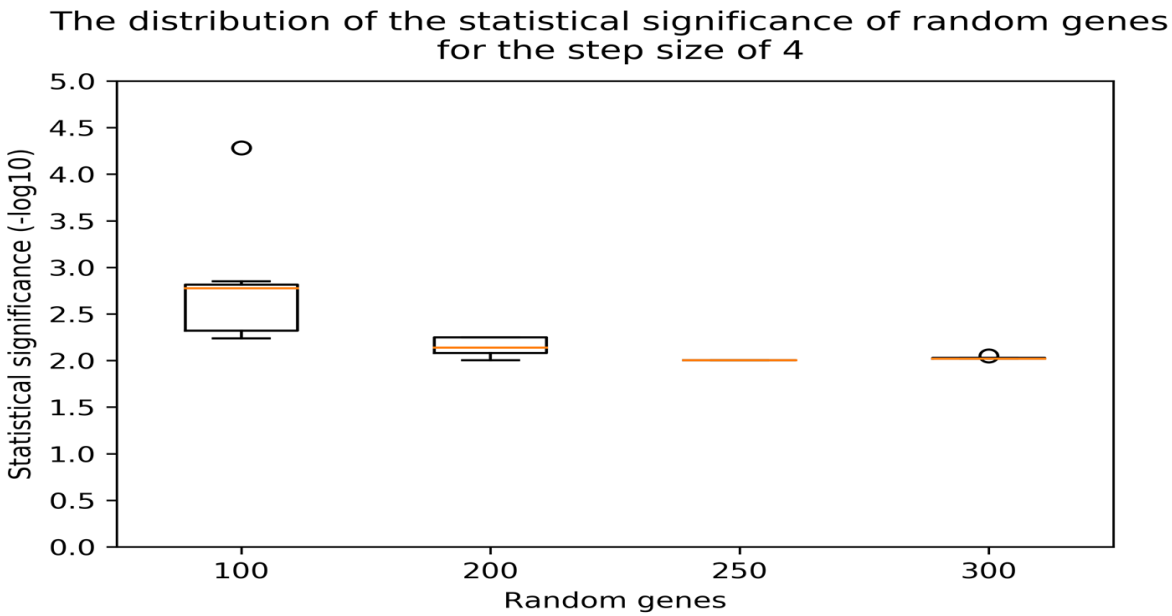


Figure 2.11: Distribution of statistical significance for random genes at step size of 4

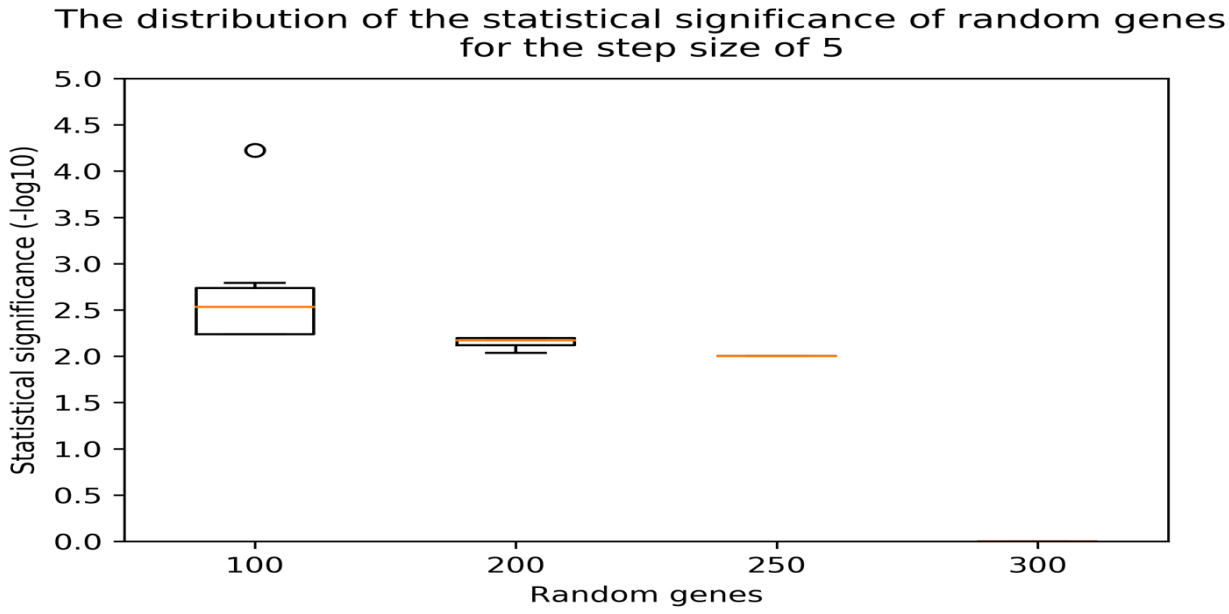


Figure 2.12: Distribution of statistical significance for random genes at step size of 5

Referring to Fig 2.8 – 2.12, it is clear that there is a systematic change in the presence of statistically significant clusters from the 300 to the 100-member data set with increase in step size. For instance, with a gap size of 1, no statistically significant clusters were observed in a random selection of 100 genes from a total pool of 1000 genes, and clusters with a statistical significance of approximately $p=10^{-2}$ was observed (Fig 2.8). With a step size of 5, no significant clusters were observed in the set of 300 random genes. However, now a number of clusters with a statistical significance of approximately $p=10^{-2.5}$ are observed (Fig 2.12). As expected, this tendency follows that of the plot shown in Fig 2.7.

A natural question arises. What is an appropriate choice for step size? As seen in the analysis above, the statistical significance is dependent on both the gap size and the size of the dataset that is analysed. An analysis of gene clustering in a genome will most likely require the analysis of a different number of gap sizes, and a perusal of the statistical significance of identified clusters, after B-H correction, to ensure that an analysis is biologically relevant.

2.7. Discussion

There are major differences between *Pyxis2* and *Cluster Locator*. *Pyxis2* calculates the exact statistical significance of a physical gene cluster based on a hypergeometric distribution and performs a false discovery rate correction on the calculated significance. *Cluster Locator*, in contrast, calculates the significance of finding the number of clustered genes as a percentage of the total number of genes for the entire data set. It provides no information on individual clusters and performs no false discovery rate correction. *Pyxis2* was also able to identify statistically significant clusters (after B-H correction) that were missed by *Cluster Locator*. I conclude that the program that I developed, *Pyxis2*, is superior in utility, accuracy and performance compared to another similar program, *Cluster Locator*.

CHAPTER 3

Domain-wide gene regulation in *S. cerevisiae*

3.1. Introduction

After the development and verification of *Pyxis2*, the program can be utilised and applied to real biological data to answer biological questions. For this study a public *S. cerevisiae* RNA-seq data set was used. The RNA-seq data set describes the level of gene expression at 15 min intervals over a 60 min period when a yeast culture re-enters the cell-cycle from stationary phase. The aim is to identify any clusters of genes that coherently become active during re-entry of the cell-cycle, and to identify the functional and evolutionary relationships between the genes in the identified clusters.

If clusters are identified, what is the likely mechanism behind the co-regulation of the genes that are clustered? Two specific possible causative signals will be investigated. The first is the possible contribution of localised changes in chromatin structure, brought about by acetylation of specific lysine residues in the N-terminal tails of histone H3 and H4. It is well established in the literature that acetylation of a number of lysine residues are associated with transcriptional activity.

The epigenome is a complex regulatory interface to the genome. A universally conserved feature, observed in all eukaryotes studied to date, is the presence of a nucleosome depleted region in the vicinity of the transcription start site. The nucleosomes on both the upstream and downstream side of this nucleosome free region contain the histone isotype, H2A.Z, which lacks an acidic domain and precludes binding of the N-terminal tail from the H4 in the adjacent nucleosome in the chromatin fibre to this region. This binding is required for the formation of condensed chromatin, and a structural feature of an H2A.Z containing nucleosome, is that the enclosing chromatin cannot condense. Thus, the chromatin is in a constitutively open state, where it is more accessible to the binding of additional protein factors. It is likely that this nucleosome depleted region is recognised by chromatin modellers such as Ino80.com and Swi/Snf that because additional nucleosome shifts in the area, also exposing binding elements that were obscured due to their close association with nucleosome surfaces. There is significant evidence that large modification complexes such as SAGA is also recruited to these regions. SAGA contains a histone acetyltransferase that will acetylate specific lysine residues in the area. These acetylated lysine residues are recognized by other bromodomain containing enzymes, which are similarly recruited to the area, preforming additional epigenetic modifications to the region. The sum total of the action of the various remodellers and modification enzymes is the formation of a region that is tagged for transcriptional activation and is in a suitably accessible conformation to allow the binding of transcriptional activators and basal factors required to stabilize the formation of the PIC.

The important point here is that the epigenetic modifications are not finely focussed. It is made in the general region, and it is possible that these modifications can also expand to surrounding regions due to the inherent amplificative nature of the molecular flags, a cycle of enzyme recruitment, and regional epigenetic modifications. It is therefore possible that the transition of one promoter area to an epigenetic state signalling transcriptional activation may well influence the neighbouring promoters to also exhibit an epigenetic activation signal.

To test this hypothesis, I plan to make use of a public ChIP-seq data set that mapped the genomic distribution of acetylated lysine during stationary phase exit and superimpose any possible clusters of acetylation sites with clusters of transcriptionally active sites to assess any possible statistically significant overlap.

Furthermore, it is also known that the nuclear architecture of chromatin is important in transcriptional activity. It is well established that Pol II transcription occurs in defined loci, and that specific compartments, close to the nuclear periphery, are associated with transcriptional repression. It is further known that the chromosomes assume a defined three-dimensional arrangement in the nucleus. Some regions of chromosomes are in close proximity and other regions are far removed. It is thus possible that transcriptionally active regions could impact on the molecular environment and transcriptional activity of regions that are in close proximity. This avenue of enquiry is specifically relevant to our current understanding of the arrangement of chromosomes in topologically isolated domains (TADs) in the nucleus, interspersed with heterochromatic regions associated with the nuclear periphery.

An alternative explanation for spatial propagation of transcriptional activity may lie in compartmentalisation. For instance, as shown in Figure 3.1, a specified compartment in the nucleus may contain a high concentration of Pol II and required ancillary basal transcription factors. Thus, parts of the chromosomes that are in close spatial proximity are likely to be within the same sub-nuclear compartment, and in chemical environments conducive to active gene expression. It is not currently possible to discriminate between site-to-site activity propagation due to proximity, and activation due to occupancy of the same sub-nuclear compartment.

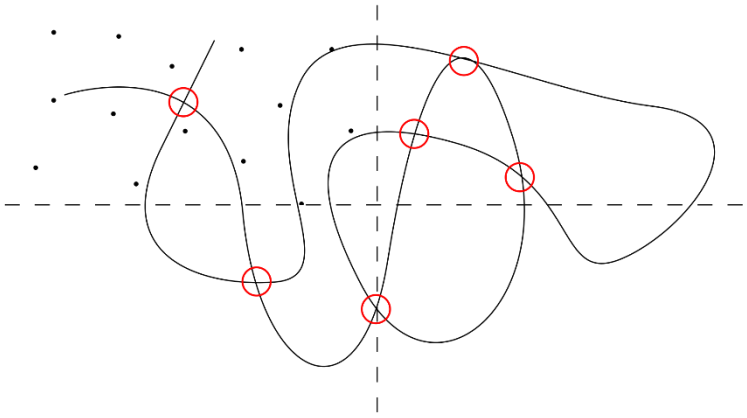


Figure 3.1. Nuclear architecture and genome function. The distribution of a single chromosome is shown, where regions where the distal regions of the same chromosome are in close spatial proximity are shown by the red circles. The compartment at the top, left of the figure, indicated with small black circles, represents an area with high concentrations of enzymes and factors required for gene expression. Areas of the genome that are in close proximity in this compartment are expected to be transcriptionally active, although it is not possible to discriminate the effect of spatial proximity from compartmentalisation with data currently available.

Nevertheless, to investigate whether nuclear proximity exhibited any effect on the gene clusters that are activated during re-entry of the cell cycle, I made use of a public Hi-C data set and assessed whether there was any statistically significant overlap between the active clusters and nuclear proximity.

3.1.1. The *S. cerevisiae* life cycle

S. cerevisiae takes approximately 100 min to double in mitotic cell division. When the cell has enough nutrients, the cell undergoes the mitotic cell cycle, and passages through the G1, S, G2 and M phases of the cell-cycle. The cells can also function in three distinct cell types (haploid a and α cells, and diploid a/α cells). If the two haploid cells mate it forms a diploid cell type, and if the diploid cell undergoes meiosis, it forms haploid cells again. The diploid cell forms an ascus which contains four spores (2 a and 2 α cells). The growth curve of *S. cerevisiae* on a carbon source has many stages such as the lag phase, log or exponential phase, stationary phase and death phase. *S. cerevisiae* can enter a non-growth stage referred to as the G_0 or stationary phase, and only exits this stage if there are sufficient nutrients available to sustain an active mitotic cell-cycle and biomass increase. Most biomass on earth survive in a nutrient deficient environment, and stationary phase is considered to be an important stage of cellular development. (79)

3.1.2. The CHIP-seq data

ChIP-seq (chromatin immune-precipitation) is a technique to study the genomic distribution of an anti-body target. This target can be any antigen that is stably bound to DNA and is accessible to an antibody. This may include a DNA-binding protein, or a modification on a DNA binding protein. An understanding of the presence of specific proteins or modifications at defined times at precise localities of the genome can provide significant insight into the role of these proteins or modifications on specific genomic processes.

A typical ChIP-seq experiment includes the chemical cross-linking of the DNA and protein of interest, fragmentation of the chromatin, immunoprecipitation of fragments containing the antibody target of interest, reversal of the chemical crosslink, purification of the DNA, followed by NGS sequencing of the recovered DNA. (80)

3.1.3. The Hi-C data

The Hi-C technique provides data on the proximity of regions of the genome. The experimental method involves crosslinking and fragmenting the chromatin, repairing DNA fragment ends by incorporating a biotinylated nucleotide, ligating the free DNA ends, removing biotin from unligated ends by exonuclease digestion, isolating the remaining biotinylated fragments with streptavidin-coated magnetic beads, and NGS sequencing of the DNA fragments. This allows the identification of the genome position of the partners in a ligated fragment, and the analysis of all such partners allows the representation of the proximity of all regions of the genome. (74)

3.2. Methods

3.2.1 Software tools used

The programming language *Python* version 3.6.1 (*Python Software Foundation, Beaverton,USA)* on the *Ubuntu* operating system version 16.04 (*Canonical Group Limited, London, UK*) was used to analyse the Hi-C, ChIP-seq and RNA-seq datasets. To generate the Hi-C matrix, the program *Juicer* was used. (81) The software *Gorilla* was used to perform gene ontology study on identified clusters to observe the enrichment of certain genes in specific identified clusters. (82) *BLASTP* searches were performed on the *National Centre for Biotechnology Information* (*NCBI*) webpage. Transcription factor binding sites in proximal promoter sequences were identified with the program AliBaba version 2.1 (http://gene-regulation.com/pub/programs/alibaba2/index.html).

Pyxis2 was used to generate data for a dataset when a *S.cerevisiae* cell exists stationary phase. The identified genes clusters were used to perform a gene ontology study to determine if there were any functional relation between the genes. A phylogenetic tree of the functionally related clusters was set up to determine if there was any duplication event that played a role in the functional relation of closely related species. ChIP-seq data on acetylated histones gathered during stationary phase exit were used to identify a possible overlap between transcriptionally active clusters and acetylated regions in the genome. Hi-C data were used to see if there was any correlation between areas of highest interaction and clusters identified. I made use of the hypergeometric distribution to calculate the possible statistical significance of overlapping clusters and acetylated or proximal genomic regions.

3.2.2. Calculating the significance of cluster overlap between two data sets

The statistical significance that the clusters of two datasets, here the RNA-seq and the ChIP-seq or Hi-C dataset, overlapped at one or more clusters, was calculated with a hypergeometric distribution. Three potential distributions, from a larger number of possible distributions, in shown in Figure 3.2.

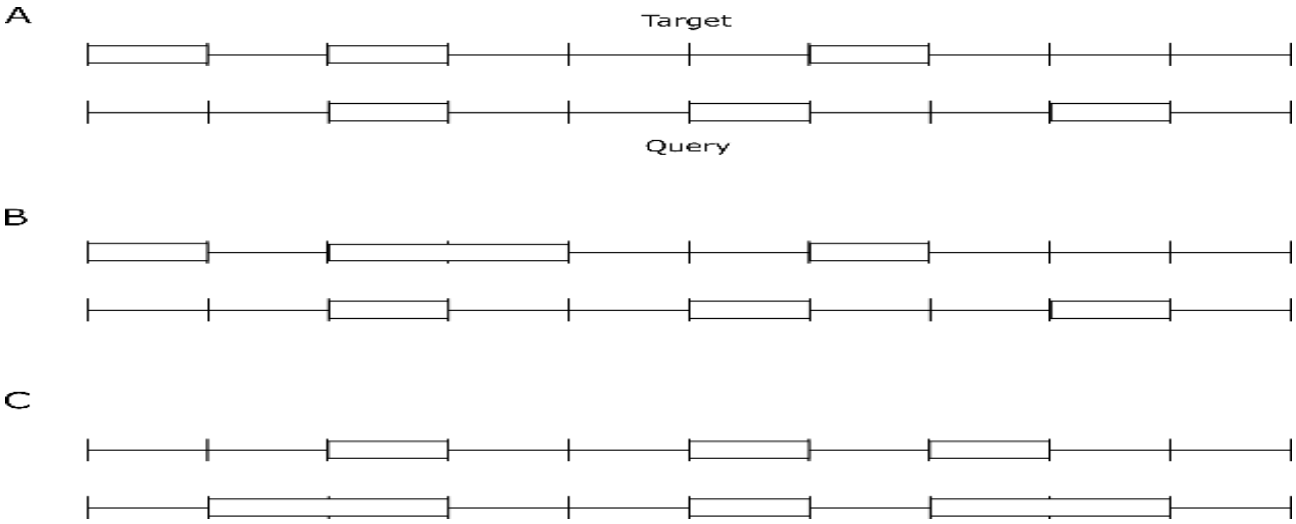


Figure 3.2: Target and query genomes represented as straight lines. The blocks indicate the different cluster region identified. The clusters are overlapped as seen in the three different segments A, B and C.

Panel A shows two random distribution of 3 clusters among 10 possible locations, in the top diagram. The hypergeometric distribution, as defined earlier, described the possibility of obtaining k successes out of a total of K possible successes, making n draws from N total possibilities. Thus, any distribution of 3 blocks over 10 possible positions is given by $\frac{1}{10} \times \frac{1}{9} \times \frac{1}{8} = 720$. However, the chance of regenerating this exact pattern by again choosing exactly the same 3 blocks, is given by $\frac{\binom{3}{3}\binom{10-3}{3-3}}{\binom{10}{3}} = 0.0083$. The chance of having at least 1 block overlap is 0.525. This is significantly larger than 0.05, indicating a significant chance that a random distribution will result in an overlap, and that such an overlap is therefore not statistically significant.

The comparison between the distributions of the clusters between the two data sets, in other words the RNA-seq and the ChIP-seq datasets, or the RNA-seq and Hi-C datasets is always made on a gene to gene basis. Thus, the distribution of the clusters can be simplified to a distribution of equally sized blocks, where each block represents a gene. The gene can either be in a cluster, or outside of a cluster. Thus, genes that are within a cluster are represented by 1 and the genes outside of the cluster by 0. Each pattern of genes in clusters is

therefore represented by a series of 0s and 1s, there the total number of digits equal the number of genes in the genome. The calculation now reduces to a comparison of the two genome datasets, and the determination of the number of genes in clusters that overlap. We make no distinction between clusters that fully overlap or only partially overlap. Thus, in genome A with 4 out of 100 genes in a cluster, and genome B with 3 genes out of a hundred genes in a cluster, what is the probability that all 3 genes in genome B are aligned with clustered genes in genome A. The probability if calculated as $\frac{\binom{4}{3}\binom{100-4}{4-3}}{\binom{100}{4}} = \frac{4\times96}{3921255} = 9.8 \times 10^{-5}$. What if the cluster sizes are unequal? Panel B and C of Figure 3.2 shows the distribution of 3 clusters, of which one is twice the size of the other two. How does this influence the calculation of significance? This contingency is not relevant to the present analysis, since we are considering individual genes, defined as either in or out of a cluster, as opposed to comparing complex clusters of variable size on different chromosomes.

3.3. Results

3.3.1. Clusters identified during stationary phase exit

This study investigated the co-expression of genes during stationary phase exit and attempted to provide a mechanistic explanation for any identified co-regulation. Cultures of *S. cerevisiae* were grown in glucose-rich medium and allowed to reach stationary phase. The cells from the cultures were transferred to a new batch of glucose-rich medium to facilitate re-entry of the cell-cycle and harvested at different time points for the microarray analysis of gene expression. (83) The expression of genes was expressed relative to time point 0, stationary phase. Genes that displayed a statistically significant induction level were used to identify clusters with *Pyxis2*. Settings for *Pyxis2* was a 4 genes step size and a cut-off p-value of 0.01. The Benjamini-Hochberg false discovery rate was applied to make the test more statistically rigorous. The code was also set to only detect clusters of genes that have a known function within the cell (no dubious ORFs). The results are represented from Figures 3.3-3.6.

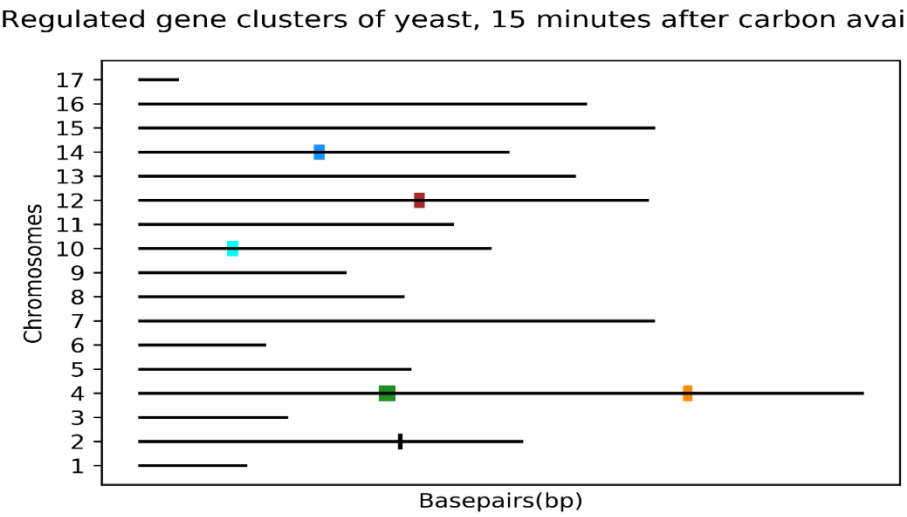


Figure 3.3: Clusters of active genes identified by Pyxis2 in S. cerevisiae cultures 15 minutes after exposure to a fermentable carbon source

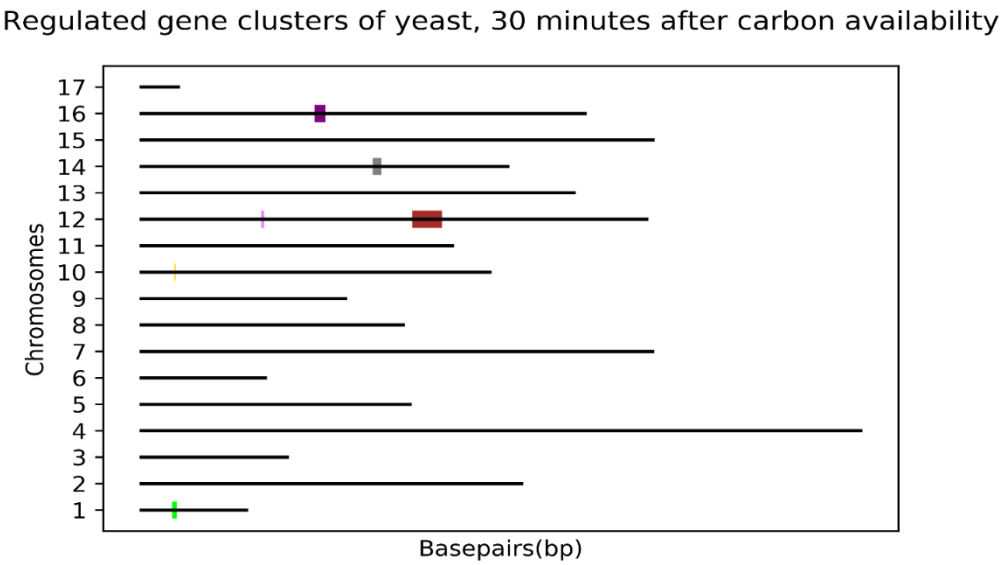


Figure 3.4: Clusters of active genes identified by Pyxis2 in S. cerevisiae cultures 30 minutes after exposure to a fermentable carbon source

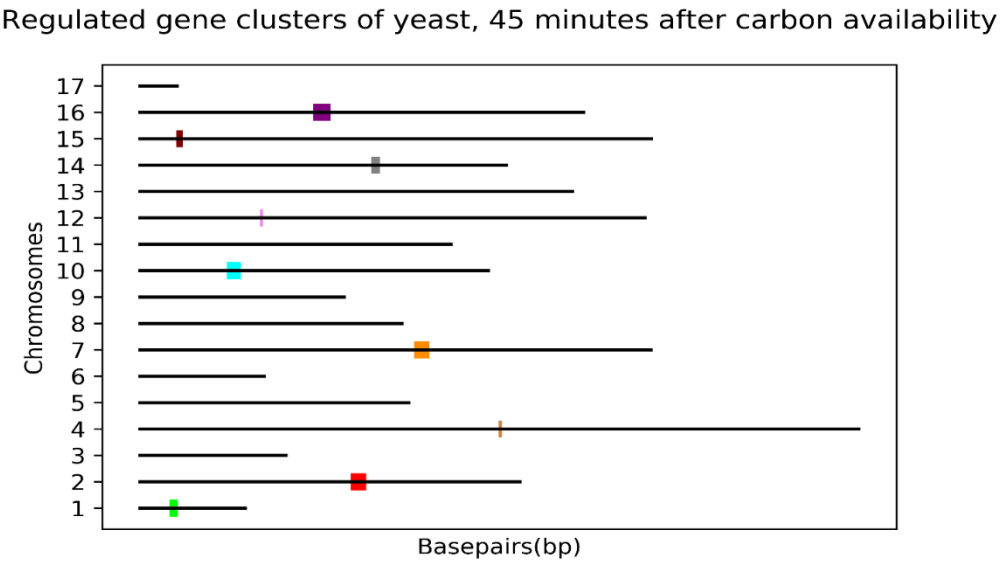


Figure 3.5: Clusters of active genes identified by Pyxis2 in S. cerevisiae cultures 45 minutes after exposure to a fermentable carbon source

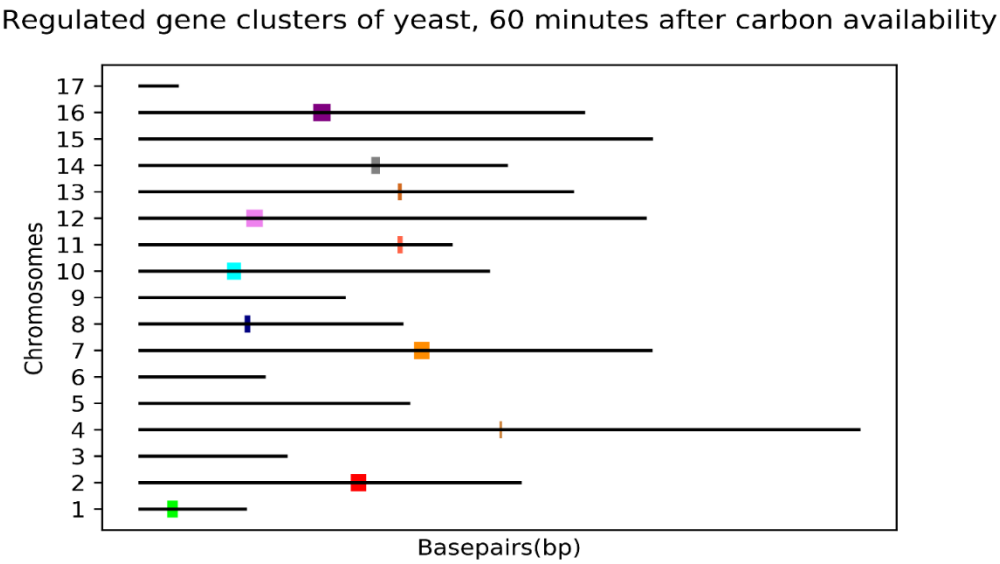


Figure 3.6: Clusters of active genes identified by Pyxis2 in S. cerevisiae cultures 60 minutes after exposure to a fermentable carbon source

Figure 3.3-3.6 shows the statistically significant clusters ($p<0.01$ followed by Benjamini-Hochberg correction) present at various time points during re-entry of the cell-cycle from stationary phase. Some clusters show very interesting dynamics. For instance, the cluster on chromosome 16 appears at 30 min, and persists at 45 and 60 min. This is clearly a cluster of genes that are switched on at a later time during stationary phase exit but persists in an active state well into the passage into the cell-cycle. Chromosome 12 exhibits a cluster that is switched on early in stationary phase exit, at 15 min, maintains an on state up to 30 min, but then becomes transcriptionally downregulated. This cluster of genes seems more associated with exit of stationary phase than with cell-cycle progression.

I next looked at the constituent genes within each cluster. The late stationary phase cluster (30, 45, 60 min) on chromosome 16 contained the genesYPL085, YPL087, YPL089, YPL090, YPL093. YPL085 is a COPII vesicle coat protein required for ER transport vesicle budding, YPL087 is an alkaline dihydroceramidase, involved in sphingolipid metabolism, YPL089 is a MADS-box transcription factor, component of the protein kinase C-mediated MAP kinase pathway involved in the maintenance of cell integrity, YPL090 is a protein component of the small (40S) ribosomal subunit, and YPL093 is a putative GTPase, associates with free 60S ribosomal subunits in the nucleolus and is required for 60S ribosomal subunit biogenesis. These genes are clearly not biochemically related, but it is feasible that they all support metabolic routes required for re-engagement of the cell-cycle.

A cluster that displayed a change in size on chromosome 12 (time point 30, 45 and 60 min region 257992 – 263954) contained the genes YLR058, YLR059, YLR060, YLR061 and YLR062. YLR058 is a cytosolic serine hydroxymethyltransferase that converts serine to glycine plus 5,10 methylenetetrahydrofolate, and is involved in generating precursors for purine, pyrimidine, amino acid, and lipid biosynthesis, YLR059 is a 3'-5' RNA exonuclease; involved in 3'-end processing of U4 and U5 snRNAs, 5S and 5.8S rRNAs, and RNase P and RNase MRP RNA, YLR060, is the beta subunit of cytoplasmic phenylalanyl-tRNA synthetase, YLR061 is the ribosomal 60S subunit protein L22A; required for translation of long 5' UTR of IME1 mRNA and meiotic entry, and YLR062 is not well-described, but may be involved in bud site selection.

Chromosome 1 contained a cluster in region 71786-79345 composed of YAL038W, a pyruvate kinase, YAL036C, a member of the DRG family of GTP-binding proteins, and YAL035W, the translation initiation factor eIF5B.

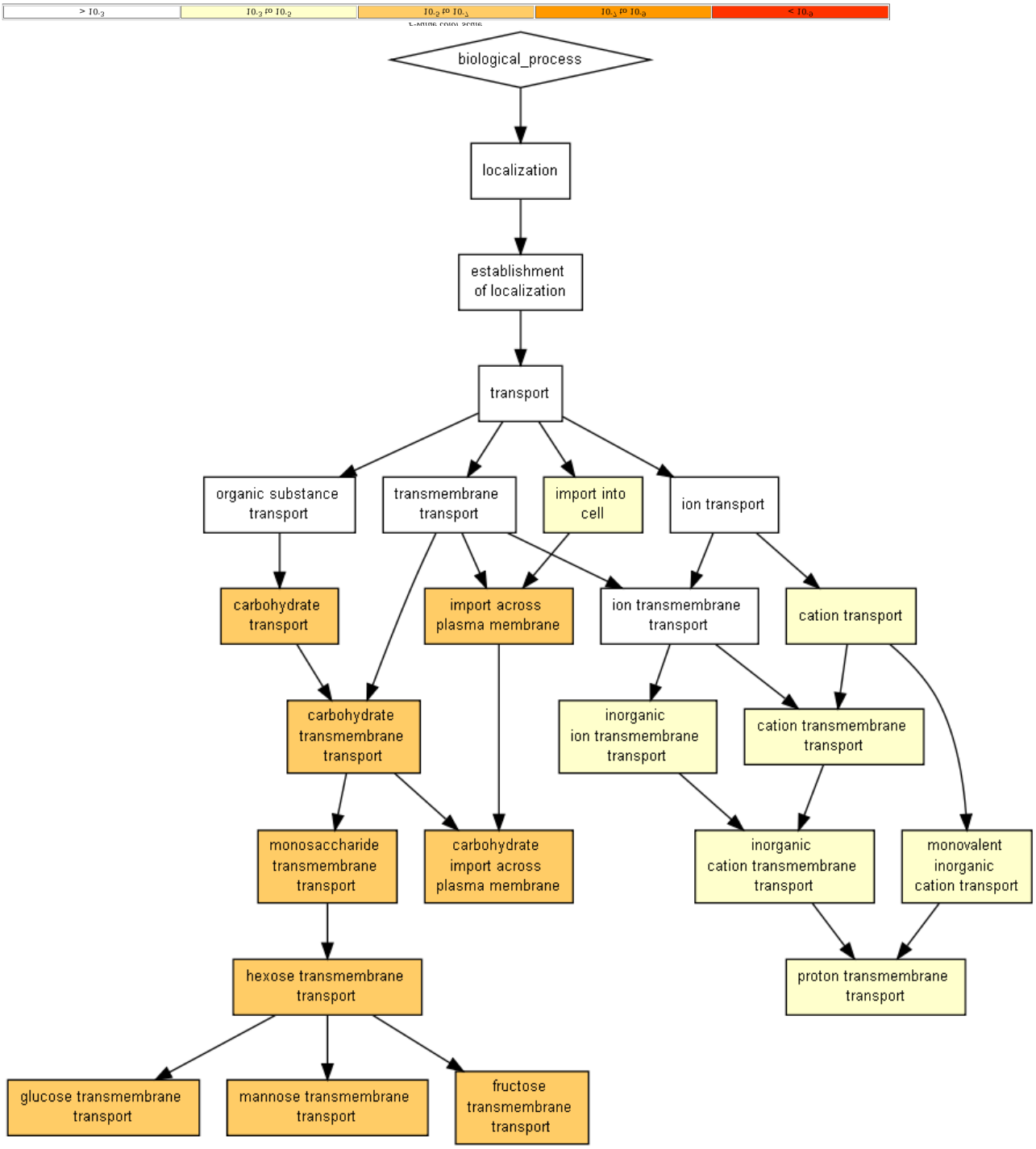


Figure 3.7. Gene ontology study of significantly enriched areas on chromosomes. The darker boxes are statistically more significant. The hexose transmembrane transport GO term enrichment showed significant enrichment with $p = 7.59 \times 10^{-7}$ *and FDR* $q=7.91 \times 10^{-4}$.

These are all examples of clusters of genes that are likely to perform essential roles during the exit of stationary phase and re-entry of the cell-cycle and may be functionally related in such a wider biological sense, but not necessarily related by catalysing reactions in a common biochemical pathway. Interestingly a cluster on

chromosome 4 that were identified at 15 min in region 1149951 - 1170143, contained genes HXT7, MVF1, SVF1, FCF1, HXT6 and HXT3. Three of the constituents' genes (HXT7, HXT6 and HXT3) are involved in glucose transport, which makes sense since the cell is re-initiating glucose import. This cluster disappears at 30 min, underscoring the dynamic nature of transcriptional induction in response to cellular state.

3.3.2. Gene ontology (GO) of clusters genes

An analysis of the ontology of the constituent genes in identified clusters was performed using *Gorilla*. (82) The clusters were individually analysed with *Gorilla* to assess whether there was any GO term enrichment. Enrichment is defined by the ratio of a GO term relative to the number of query genes compared to the ratio of all genes with the specific GO term relative to all genes in the genome.

The cluster on chromosome 4 (15 min) was identified with significant GO term enrichment (Figure 3.7). As discussed above, three of these gene products HXT3, HXT6 and HXT7 are involved in glucose transport. HXT 6 and HXT7 are virtually identical, differing at a single residue at position 556 (alanine to threonine substitution). HTX3 is evolutionary more diverse, with significant differences in the N-terminal region. The multiple alignment of HXT3, HXT6 and HXT7 is shown is Figure 3.8.

```
HXT6          --MSQDAAIAEQTPVEHLSAVDSASHSVLSTPSNKAERDEIKAYGEGEEHEPVVEIPKRP
HXT7          --MSQDAAIAEQTPVEHLSAVDSASHSVLSTPSNKAERDEIKAYGEGEEHEPVVEIPKRP
HXT3          MNSTPDLISPQKSSENSNADLPSNSSQVMNMPEEKGVQDDFQAEAD-----QVLTNPNTG
                    : *   .::::. :  : : * * .*:. *.:*. :*:::* .:      *:  *:

HXT6          ASAYVTVSIMCIMIAFGGFVFGWDTGTISGFINQTDFIRRFGMKHKDGTNYLSKVRTGLI
HXT7          ASAYVTVSIMCIMIAFGGFVFGWDTGTISGFINQTDFIRRFGMKHKDGTNYLSKVRTGLI
HXT3          KGAYVTVSICCVMVAFGGFVFGWDTGTISGFVAQTDFLRRFGMKHKDGSYYLSKVRTGLI
                  .*******     *:*:*****************:     ****:**********:
**********

HXT6          VSIFNIGCAIGGIILSKLGDMYGRKVGLIVVVVIYIIGIIIQIASINKWYQYFIGRIISG
HXT7          VSIFNIGCAIGGIILSKLGDMYGRKVGLIVVVVIYIIGIIIQIASINKWYQYFIGRIISG
HXT3          VSIFNIGCAIGGIILAKLGDMYGRKMGLIVVVVIYIIGIIIQIASINKWYQYFIGRIISG

***************:*********:**********************************

HXT6          LGVGGIAVLSPMLISEVSPKHLRGTLVSCYQLMITAGIFLGYCTNFGTKNYSNSVQWRVP
HXT7          LGVGGIAVLSPMLISEVSPKHLRGTLVSCYQLMITAGIFLGYCTNFGTKNYSNSVQWRVP
```

```
HXT3          LGVGGIAVLSPMLISEVAPKEMRGTLVSCYQLMITLGIFLGYCTNFGTKNYSNSVQWRVP
                 *****************:**.:*************
************************

HXT6          LGLCFAWALFMIGGMTFVPESPRYLAEVGKIEEAKRSIAVSNKVAVDDPSVLAEVEAVLA
HXT7          LGLCFAWALFMIGGMTFVPESPRYLAEVGKIEEAKRSIAVSNKVAVDDPSVLAEVEAVLA
HXT3          LGLCFAWALFMIGGMTFVPESPRYLVEAGQIDEARASLSKVNKVAPDHPFIQQELEVIEA
                 *************************.*.*:*:**: *::  **** *.* :  *:*.:
*

HXT6          GVEAEKLAGNASWGELFSSKTKVLQRLIMGAMIQSLQQLTGDNYFFYYGTTIFKAVGLSD
HXT7          GVEAEKLAGNASWGELFSSKTKVLQRLIMGAMIQSLQQLTGDNYFFYYGTTIFKAVGLSD
HXT3          SVEEARAAGSASWGELFTGKPAMFKRTMMGIMIQSLQQLTGDNYFFYYGTTVFNAVGMSD
                 .**             :      **.*******:.*.      :::*      :**
*********************:*:***:**

HXT6          SFETSIVLGIVNFASTFVGIYVVERYGRRTCLLWGAASMTACMVVYASVGVTRLWPNGQD
HXT7          SFETSIVLGIVNFASTFVGIYVVERYGRRTCLLWGAASMTACMVVYASVGVTRLWPNGQD
HXT3          SFETSIVFGVVNFFSTCCSLYTVDRFGRRNCLLYGAIGMVCCYVVYASVGVTRLWPNGEG
                 *******:*:***     **          .:*.*:*:***.***:**     .*..*
***************:.

HXT6          QPSSKGAGNCMIVFACFYIFCFATTWAPIPYVVVSETFPLRVKSKAMSIATAANWLWGFL
HXT7          QPSSKGAGNCMIVFACFYIFCFATTWAPIPYVVVSETFPLRVKSKAMSIATAANWLWGFL
HXT3          NGSSKGAGNCMIVFACFYIFCFATTWAPIAYVVISETFPLRVKSKAMSIATAANWLWGFL
                 :
***************************.***:**************************

HXT6          IGFFTPFITGAINFYYGYVFMGCLVFMFFYVLLVVPETKGLTLEEVNTMWEEGVLPWKSA
HXT7          IGFFTPFITGAINFYYGYVFMGCLVFMFFYVLLVVPETKGLTLEEVNTMWEEGVLPWKSA
HXT3          IGFFTPFITGAINFYYGYVFMGCMVFAYFYVFFFVPETKGLTLEEVNDMYAEGVLPWKSA
                 ************************:**     :***::.*************     *:
*********

HXT6             SWVPPSRRGANYDAEEMAHDDKPLYKRMFSTK
HXT7             SWVPPSRRGANYDAEEMTHDDKPLYKRMFSTK
HXT3             SWVPTSQRGANYDADALMHDDQPFYKKMFGKK
```

```
                ****.*:*******: : ***:*:**:**..*
```

Figure 3.8. Multiple alignment of HXT3, HXT6 and HXT7. Conserved and identical residues are indicated by colon and asterisk symbols. The single residue difference between HXT6 and HXT7 is underlined.

```
HXT6        --------------TGGCATCAAATTTGGGAAAATTTAGCTTACTCAGACACCAACACAT
HXT7        ----------------------------AATAGTACTCTCATCGCTAAGATCATTTGGGGT
HXT3        ACCGGTATATCAAATGGCGGTGTAGTTTGAAAAGTACACTGTATGTCCATTAATAAAATT
                                         *               *         *

HXT6        TTTTTCATGAGTATTG-TTGG--AAA-ATCTTTCCATGCTTATCAGCCGGAATACGGAAC
HXT7        TGTTAAGCATGCCCTG-CTAA--ACACGCCCTACTAAACACTTCAAAAGCAACTTAAAAT
HXT3        ACGCCGAAGAACACTGACTAGCTATAAATTCTCTTCCGGCGTCCAATTTCAACCTAAGGC
                         **  *    * *     *           **     **

HXT6        AACGGCTGCTGT--GATGCCGCAAAGGGGCAAGGAAAAAATAAGCAACACACACCGAAAC
HXT7        ATTT-TTATCTA--ATTATAGCTAAAACCCAATGTGAAAG-ACATATCATACTGTAAAA-
HXT3        AAGTATTGTTAATCAATAATGCGTGTTGGTAAATATGAAGCCCGATGTTGATTAAGAAGA
               *     *         *   **        **     **           *     **

HXT6              AGGA-----GGAAGAAATTCCGAACACAAGCCTAAGAAGACACCGCAGTTG--
TTGCAAC
HXT7        GTGA-----AAAAGCAGCACCGTTGAACGCCGCAAGAGTGCTCCCATAACGCTTTACTAG
HXT3        ATTATTCGTATAAGAATTAAGCAAGCAAAATTAAGGAAAATTTTTTCTTTCCTATTCGTC
                  *       *** *                 * **                 * *

HXT6          ATGTTCAC--ATGTTCATCCCCCTGCTACTACTTGGAAATTAATGTAGGGGTAATTG-
--
HXT7        AGGGCTAG--ATTTTAATGGCCCCTTCATGGAGAAGTTATGAGGACAAATCCCACTACA-
HXT3        ATCGCAGACAGCCTTCATCTTCTCGAGATAACACCTGGAGGAGGAGCAATGAAATGAAAG
               *            ** **   *     *          *  *           *

HXT6        GAGTGCGCCGGCCCGTGCTTTCAGAGGAACAAAGGAAGAATTGTTAAAAAAAGGCAGTGA
HXT7        GAAAGCGCAACAAATTTTTTTTTCCGTAACAACAAACATCTCATCTAGTTTCTGCCTTAA
HXT3        GAAAAAAAAATACTTTCTTTTTCTTGAAAAAAGAAAAAAATTGTAAGATGAGCTATTCGC
               **            *  ***    * ** **   *    *  *
```

```
HXT6          CAACGAGCACGGGTGCACCATATGGTGTGATA-AGCAACCTAGGACAAAAAGAAGTGCGC
HXT7          ACAAAGCCGCAGCCAGAGCCGTTTTTCCGCCATATTTATCCAGGATT----GTTCCATAC
HXT3          GGAACATTCTAGCTCGTTTGCATCTTCTTGCA---TTTGGTAGGTTTT---CAATAGTTC
                   *        *          *  *     *         ***              *

HXT6                  AGTTCCTGTGTGCGGCGCCAAGACAAATGTTTCATATGCTTCTCCAAG----
GGGCTACG
HXT7          GGCTCCGTCAGAGGCTGCTACGGGATGTTTTTTTTTTACCCCGTGGAAATGAGGGGTATG
HXT3          GGTAATATTAACGGATACCTACTATTATCCCCTAGTAGGCTCTTTTCACGGAGAAATTCG
                  *            *   *          *              *           *   *
*

HXT6          CT---ACCTAGCCTCACCACCCGATCTCTTTTTTTCCTGAACGCAGAGGGGACCGTACGA
HXT7          CAGGAATTTGTGCGGGGTAGGAAATCTTTTTTTTTTTTAGGAGGAACAACTGGTGGAAGA
HXT3          GGAGTGTTTTTTTTCCGTGCGCATTTTCTTAGCTATATTCTTCCAGCTTCGCCTGCTGCC
                        *                * * **   *   *      *         *

HXT6          AAAAAAATGTTTTTTAGGCAACGGAGATTCGTTTTATCCACGTTTACCCCACAAAAAGTG
HXT7          ATGCCCACACTTCTCAGA-AATGCA--TGCAGTGGCAGCACGCTAATTCGAAAAAATTCT
HXT3          CGGTCATCGTTCCT---GTCACGTAGTTTTTCCGGATTC--GTCCGGCTCATATAATACC
                        *  *      * * *  *          *  *        * * **

HXT6          CAGGTA---CATTGTGGGGCCCCGGCATCGAAAACCAGTTTTTTTCCTTTAAACGCTGGA
HXT7          CCAGAAAGGCAACGCAAAATTTTTTTTCCAGGGAATAAACTTTTTA--TGACCCACTACT
HXT3          GCAATA----AACACGGAATATCTCGTTCCGCGGATTCGGTTAAACTCTCGGTCGCGGAT
                       *    *                *          **      *    * *

HXT6          AAAAAAGGAGAAATTATTGGAACTTTGCAGAGAATAGTCCGTAGG--CAAATTGAAAATG
HXT7          TCTCGTAGGAACAATTTCGGGCCCCTGC---GTGTTCTTCTGAGGTTCATCTTTTACATT
HXT3          TATCACAGAGAAAGCTTCGTGG--------AGAATTTTTC-------CAGATTTTCCGCT
                        *  * *   * *           *  *  * *       **  **

HXT6          TTCCTTAAAAAATTTCGTTTCTTACTCATTGAG-ATTATTCAGATGCCCTCCGTGCCTTC
HXT7          TGCTTCTGCTGGATAATTTTCAGAGGCAACAAGGAAAAATTAGATGGCAAAAAGTCGTCT
HXT3               TTCCCCGATGTTGGTATTTCCGGAGGTC------ATTATACTGA-----------
CCGCC
                   * *             ** *  *          *  *    **          *
```

```
HXT6                ATTGAA-AAAAATCCAAGAGATGTCTCGGATCTGTATGCAGATTTTGGCTTGC--
AGACA
HXT7          TTCAAGGAAAAATCCCCACCATCTTTCGAGATCCCCTGTAACTTATTGGCAACTGAAAGA
HXT3               ATTATAATGACTGTACAACGACCTTCTGGAGAAAGAAACAACTCAATAACGAT--
GTGGG
                       *       *          *  *   *           *  *

HXT6          ATGGAGAGCAAATGGGTATACAATATAGAAAGCACAGAAACATATAAAAAGAGCTCGAGA
HXT7          ATGAAAAGGAGGAAAATACAAAATATACTAGAACTGAAAAAAAAAAAGTATAAATAGAGA
HXT3          ACATTGGGGGCCCACTCAAAAAATCTGGGGACTATATCCCCAGAGAATTTCTCCAGAAGA
                 *      *         * * *** *              * * **         ***

HXT6           AAAGACATAT-GGTTTGTAACTATCTTCT--TCTTTTTTCCAATTTTT-CTGTTTTAA-
-
HXT7          CGATATATGCCAATACTTCACAATGTTCGAATCTATTCTTCATTTGCAGCTATTGTAAAA
HXT3          GAAGAAAAGTCAAAGTTTTTTTTCGCTTGGGGGTTGCATATAAATACAGGCGCTGTTTTA
                   * * *          *        *      *    *  *  *        * *

HXT6          TAATAAAAAAACAAGAACAAACAAGCTCAACTTGTCTTTTCTAAGAACAAAGAATAAACA
HXT7          TAATAAAACATCAAGAACAAACAAGCTCAACTTGTCTTTTCTAAGAACAAAGAATAAACA
HXT3          TCTTCAG-CATGAATATTCCATAATTTTACTTAATAGCTTTTC---ATAAATAATAGAAT
                 *  * *   *  ** *    * **  * *  *  *   ** *    * *** ****
*

HXT6             CAAAAACAAAAAGTTTTTTTAATTTTAATCAAAAA
HXT7             CAAAAACAAAAAGTTTTTTTAATTTTAATCAAAAA
HXT3             CACAAACAAAATTTACATCTGAGTTAAACAATC--
                 ** ********  *   * * * ** **  *
```

Figure 3.9. Multiple alignment of the 1000 bp sequences upstream of the translation start sites of HXT3, HXT6 and HXT7. Note the relative conservation in the region approximately spanning the 50 nucleotides directly upstream of the translation start site (underlined).

The degree identity between HXT3, HXT6 and HXT7 suggests that this series of genes may have originated by gene duplication events. However, the degree identity is likely irrelevant to the co-regulation of these genes. It would be much more informative to examine the proximal promoter areas of these three genes.

The multiple alignment of the 1000 bp upstream of the translation start codon of HXT3, HXT6 and HXT7 show that the proximal promoter areas, with the exception of the approximately 50 bp present directly upstream of the translation start site, are quite diverse (Figure 3.9).

I next identified all transcription factor binding sites present in the 1000 bp proximal promoter upstream regions of HXT3, HXT6 and HXT7, as well as all sites present upstream of all three genes. The result is given in Table 3.1

Table 3.1. Transcription factor binding sites present in the 1000 bp upstream of the HXT3, HXT6 and HXT7 genes. Note that not all transcription factors occur in *S. cerevisiae*. Factors indicated by green boxes bind to sites identified upstream of each of the three transported genes.

Transcription Factor			**Description**
HXT3	**HXT6**	**HXT7**	
1-Oct	1-Oct	1-Oct	
11-Oct	Adf-1	11-Oct	
Antp	Antp	Adf-1	
AP-1	AP-1	Antp	
AP-2alphaA	C/EBP	AP-2	
ATF-a	C/EBPalpha	C/EBP	
C/EBP	C/EBPdelta	C/EBPalpha	
C/EBPalpha	c-Rel	C/EBPbeta	
C/EBPbeta	Dl	C/EBPdelta	
c-Fos	E2	CeMyoD	
c-Jun	Erg-1	c-Jun	
COUP	GAL4	CRE-BP1	
CPE_binding_pro	GATA-1	E2	
CREB	GCN4	Elf-1	
c-Rel	GLO	Elk-1	
dioxin_receptor	GR	Erg-1	
Ftz	Hb	Ftz	
GABP	HNF-1	GATA-1	
GATA-1	HNF-1C	GCN4	Transcriptional activator of nitrogen catabolite repression genes
GCN4	HNF-3	GLO	bZIP transcriptional activator of amino acid biosynthetic genes; activator responds to amino

			acid starvation
GR	HNF-3B	GR	
Hb	HOXA4	Hb	
HEB	ICSBP	HNF-1	
HNF-1	IRF-1	HNF-1C	
HNF-1C	LyF-1	HNF-3	
HNF-3	MEB-1	HNF-3B	
HSE-binding_pro	MIG1	HOXA4	
HSF	NF-1	ICSBP	
HSTF	NF-EM5	MEB-1	
ICSBP	NF-kappaB	NF-1	
MAL63	Oct-1A	NF-EM5	
MCM1	Oct-2.1	NF-kappaB	
MIG1	Pap1+	NF-kappaB1	
MyoD	Pit-1	NRL	
NF-1	PU.1	Oct-1A	
NF-kappaB	repressor_of_CA	Odd	
Odd	Sp1	RSRFC4	
p40x	TBP	SGF-1	
Pit-1	Zen-1	Sp1	
Pit-1a		SRF	
RSRFC4		TAF-1	
Sp1		TBP	
SRF		TEC1	
TBP		Zen-1	TATA-binding protein (TBP); general transcription factor that interacts with other factors to form the preinitiation complex at promoters

As can be seen in Table 3.1, several transcription factor sites are present upstream of all three genes. Note that not all transcription factor binding sites are paired with factors that are present in yeast. The HNFx factors, for instance, are hepatocyte nuclear factors that were identified in humans, and do not occur in *S. cerevisiae*. However, GATA-1, GCN4 and TBP are all factors found in yeast. Both GATA1 and GCN4 are striking, since these factors respond to nitrogen catabolite repression and amino acid starvation. These are regulatory proteins expected to be upregulated during exit of starvation-induces stationary phase.

3.3.3. Phylogenetic tree of functionally enriched clusters

An important question is whether the genes that are co-regulated in a cluster in *S. cerevisiae* are similarly conserved as a cluster in evolutionary related species, or whether the neighbouring positions in the genome is simply a random evolutionary event observed on *S. cerevisiae*. Paralogues of each of the genes observed in the above cluster on chromosome 14 were identified in other yeast species with BLASTP. The identity between the entire cluster sequence and that of other yeast species is shown in Figure 3.10. It is clear from this figure that the fully defined cluster already existed in *S. kudriavezvii*. This cluster formed after the whole genome duplication event in the *Saccharomyces* lineage some 100-200 million years ago. It was lost in the *S. paradoxus* and *S. mikatae* lines, but conserved in the *S. cerevisiae* line.

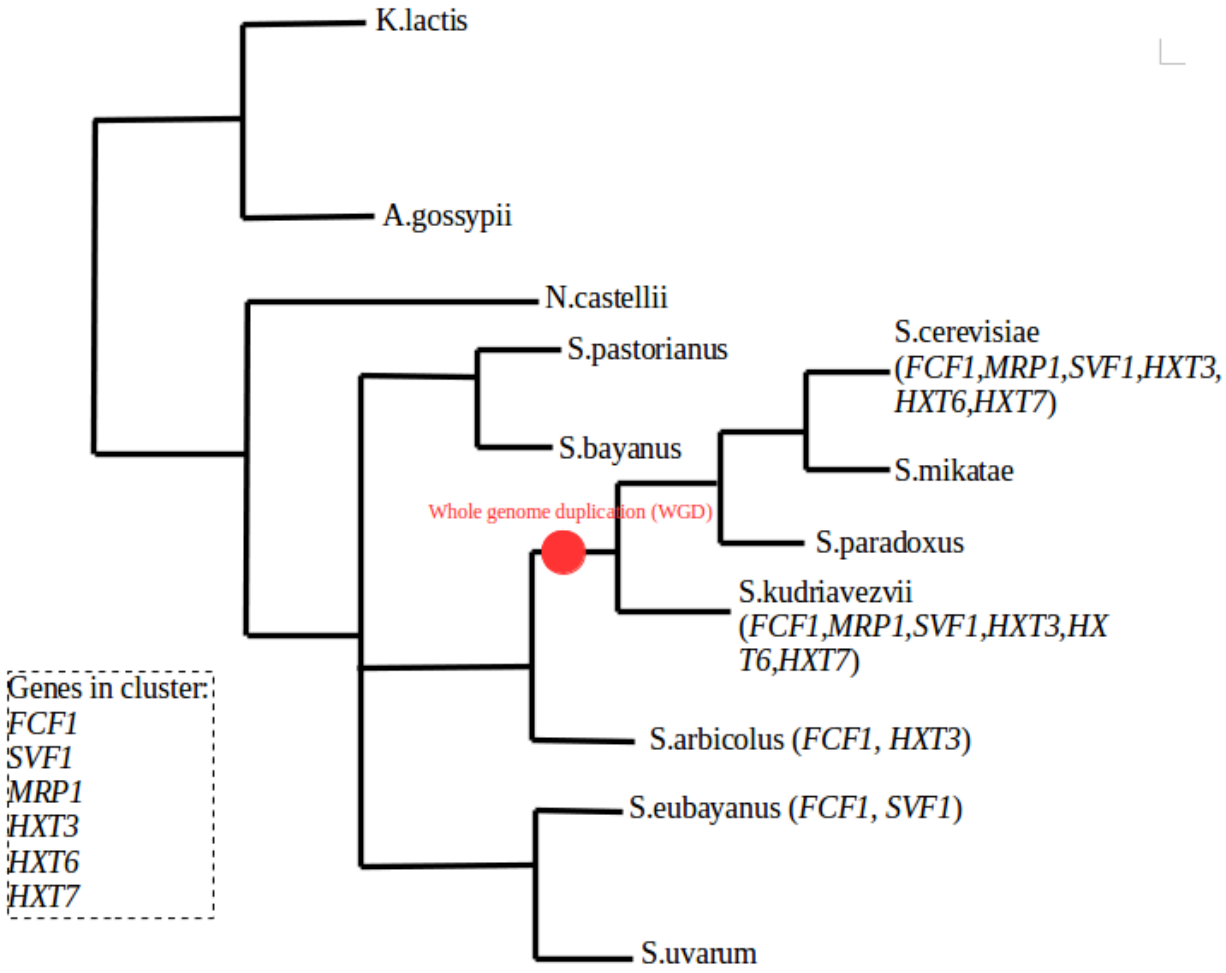


Figure 3.10. Phylogenetic tree related to enriched cluster and that are found in closely related species

The genes shown in Figure 3.10 had a percentage identity >95% to be defined as paralogues. Two genes in the cluster were identified in the species *S. eubayanus* (FCF1 and SVF1) and *S. arbicolus* (FCF1 and HXT3).

Whole genome duplication (WGD) in *S. cerevisiae* has evolutionary advantage since it neutralises deleterious mutations. (84) Most eukaryotic species show evidence of WGD, and it was proposed that the WGD in yeast did not simply involve the doubling of its DNA, but also the hybridization of genetic material from a different species. (85)

3.3.4. Acetylation of chromatin domains

3.3.4.1 SAGA complex

The overlap of genes in clusters between the RNA-seq dataset, containing genes that are upregulated at distinct times during re-engagement of the cell-cycle, and the SAGA ChIP-seq dataset, containing a list genes associated with significantly enriched levels of the SAGA acetyltransferase complex, was compared, and the statistical significance of overlap of clustered genes calculated for each chromosome and time point independently. The results are presented in Table 3.2.

Table 3.2: Overlap of SAGA Complex with Pyxis 2 data

Time point (min)	**Positions of overlap**	**Parameters (M,n,N,k)**	**Significance (<0.05)**
30	**Chr 1 – 20,21,22**	**78,10,3,3**	**0.00157**
60	Chr 1 – 20,21	78,15,3,2	0.0869
60	Chr 2 – 45	348,12,4,2	0.125
60	**Chr 7 – 58,60,62**	**454,18,5,3**	**0.000492**
30	Chr 10 - 7	302,16,1,1	0.0530
60	Chr 10 - 19	302,20,4,1	0.218
60	Chr 13 - 55	385,30,1,1	0.0780
30	**Chr 14 – 49,50,53,57,59**	**335,41,5,5**	**2.197e-05**
60	Chr 16 - 40	397,10,4,1	0.0940

M = total number of blocks

n = number of highly reactive blocks in target genome

N = number of reactive blocks in query genome (number of attempts)

k = number overlapping blocks from query genome to target genome (number of successes)

There was significant overlap of transcriptionally induces genes, and genes associated with SAGA at time point 30 for both chromosome 1 and 14 and also time point 60 for chromosome 7. The statistical significance was well in excess of 95% significance. This result therefore strongly suggests that there is a correlation

between the SAGA binding and gene activation. However, it should be stressed that this is clearly not true for all upregulated genes, or all genes associated with the SAGA complex. This discrepancy can most likely be ascribed to temporal dynamics and redundancy. It is possible that SAGA may have been involved in the acetylation of genes in early stationary phase exit, which subsequently become transcriptionally active, and may form part of a co-regulated cluster. The early binding of SAGA may have been missed in the ChIP-seq experiment. Alternatively, SAGA is not the only histone acetyltransferase complex involved in transcriptional activation. To gain a more complete insight into the relationship between gene acetylation and transcriptional activation, I analysed the acetylation state of K9 of histone H3 directly.

3.3.4.2. H3K9ac

The overlap between the genes forming clusters in the RNA-seq dataset and the H3K9ac dataset is shown in Table 3.3. Curiously, no overlap in clusters were found. It is not clear whether H3K9ac takes place after acetylation of other lysine residues, notable H3K14 and H3K27. Both these modifications have been shown to be associated with a state of transcriptional activation. Thus, although we have shown an association with SAGA at some transcriptionally active clusters, it does not seem that H3K9ac is a modification that is associated with active gene clusters during exit of stationary phase in yeast.

Table 3.3: Overlap of H3K9ac with *Pyxis2* data

Time point (min)	**Positions of overlap**	**Parameters (M,n,N,k)**	**Significance (<0.05)**
60	Chr 2 - 45	348,14,4,1	0.143
60	Chr 8 - 24	236,7,2,1	0.0507
30	Chr 10 – 7	302,23,1,1	0.108
60	Chr 10 - 19	302,14,4,1	0.162
60	Chr 11 - 55	279,18,3,1	0.314
60	Chr 12 - 55	431,21,7,1	0.256
30	Chr 16 - 37	302,20,3,1	0.174
60	Chr 16 - 37	302,15,4,1	0.172

M = total number of blocks

n = number of highly reactive blocks in target genome

N = number of reactive blocks in query genome (number of attempts)

k = number overlapping blocks from query genome to target genome (number of successes)

3.4. Hi-C data

I was next interested to investigate the possibility that regions of the genome that were in close physical proximity, were also associated with the transcriptionally active gene clusters. The overlap between spatially proximal regions in normally cycling yeast cells and yeast cells in stationary phase was determined by Hi-C. I obtained the data set from the public domain and analysed the possible overlap of the genes in clusters in the RNA-seq data set and the Hi-C data set. The results are shown in Table 3.4

Table 3.4. Overlap of Hi-C data to *Pyxis2* data

Condition	**Positions of overlap**	**Parameters (M,n,N,k)**	**Significance (<0.05)**
Normal	**Chr 12 – 87,89,92,93,94,97,100, 102,103**	**431,65,10,10**	**3.263e-09**
Glucose starved	Chr 12 – 0	431,31,10,0	No value

M = total number of blocks

n = number of highly reactive blocks in target genome

N = number of reactive blocks in query genome (number of attempts)

k = number overlapping blocks from query genome to target genome (number of successes)

Referring to Table 3.4, it is seen that a total of 10 clusters were found to overlap between spatially proximal regions of the genome and transcriptionally active gene clusters during stationary phase exit. This correlation was only seen in normally cycling cells, however. The Hi-C analysis of the stationary phase genome did not show any statistically significant correlation. This may mean that transcriptionally active gene clusters are taken up into spatially proximal regions of the genome during normal cycling, and that the spatial proximity in the genome is not a causative signal for the formation of transcriptionally active clusters, but that the reverse is true: transcriptionally active clusters define regions that subsequently become spatially proximal in the genome.

3.5. Conclusion

Some of the gene clusters identified during exist of stationary phase contain genes involved in glucose transport. Some of the genes in this cluster were formed by the WGD event, since a phylogenetic analysis indicated similar, but not identical genes in closely related species. There was significant overlap of clusters of the functional genomics data with the SAGA complex, but not any significant overlap for H3K9ac. I conclude that H3K9ac is not an epigenetic mark associated with gene activation during cell-cycle re-entry. It is possible that acetylation of other lysine residues, such as K14 and K27 of H3 are involved in stationary phase exit gene activation. The Hi-C data for glucose fed *S. cerevisiae* overlapped with 2 gene clusters on

chromosome 12. These clusters did not exhibit a statistically significant overlap with spatially proximal regions in the stationary phase genome. However, when compared to the genome of a normally cycling cell, significant overlap of 6 clusters on chromosome 12 was observed. This result suggests that, contrary of expectation, close spatial distances is not a causative effect in activation of clusters of active genes in stationary phase exit, but that active clusters may, in fact, induce close spatial proximity in the genome.

CHAPTER 4
Discussion and conclusion

4.1. Discussion

A bioinformatics tool was developed to determine the statistical significance of physical clusters of genes in any functional genomics dataset for an organism for which a GFF3 genome annotation file is available. *Pyxis2* was validated using synthetic datasets. New developments were made to the program compared to the previous developed *Pyxis*. Examples include the agnostic GFF3 file that *Pyxis2* can incorporate, and also the fact that the program can be applied on large genomes. The hypergeometric distribution was incorporated to validate the detected clusters and the program was compared to a similar program, *Cluster Locator*. *Pyxis2* was found to be more sensitive than *Cluster Locator* in the detection of protein coding genes. This is because *Cluster Locator* includes dubious and merged ORFs in their results. A variable parameter for *Pyxis2* is the step size, and it was found that the optimal step size was between 2 to 4.

Further analysis was done on microarray data set as the cell exists stationary phase. See appendix for the BED files output of the cluster regions. There were several clusters that were continuously up regulated, and some that re-appeared or disappeared after a certain time period. It was expected that genes involved in glucose uptake should be induced during stationary phase exit, because the cells are induced to re-enter the cell-cycle by providing glucose in fresh growth media. This expected upregulation was observed. HXT7, HXT6 and HXT3, all involved in glucose uptake, were induces in a single cluster.

The HXT7, HXT6 and HXT3 cluster may have appeared after the WGD event and the paralogous genes were found in the genome of *S. kudriavezvii*, closely related to *S. cerevisiae*, and an evolutionary offshoot that occurred after the WGD event. This implies that the cluster may have evolved after the WGD event. Curiously, both *S. paradoxus* and *S. mikatae* lack the gene cluster, even though these two species share the same ancestor as *S. kudriavezvii.* It would be interesting to study the genomes of *S. paradoxus* and *S. mikatae* for structural evolutionary events, to understand how the cluster was lost. It will be equally interesting to understand which genes are involved in glucose transport in these two species.

ChIP-seq and Hi-C data was used to answer the question of whether the spatial structure of the genome has an effect on the genome-wide regulation of clusters. This might give insight into how these clusters are co-regulated. Comparing clusters identified during stationary phase exit to clusters identified in the ChIP-seq data gave information on the activation of gene clusters. The acetylation of histones causes the chromatin to adopt an open conformation and make the entry of transcription factors to certain areas of chromatin possible. The ChIP-seq analysis indicated that there was more overlap in the binding of the SAGA complex activated gene clusters, than between acetylated H3K9 regions and active clusters.

I speculate that SAGA, which is recruited to the TATA box regions of genes, is either very transiently associated with genes about to be activated or may bind to very few genes in a cluster to be the SAGA enriched genes, and all genes that constitute a cluster, determined from the RNA-seq data. It remains possible that an early association of SAGA with some of the genes in the cluster can lay down the activation signal, which is amplified through the region by subsequent binding of bromodomain containing acetyltransferases. However, there clearly was very little overlap of H3K9ac, the target of SAGA, and the activated gene clusters. This may mean that other acetyltransferases are involved in upregulating transcription of the active clusters, such as acetyltransferases that target H3K14 and H3K27. There were no data sets available on H3K14ac and H3K27ac during stationary phase exit, so it was not currently possible to address this possibility. Nevertheless, the statistically significant overlap observed for some SAGA clusters and activated clusters do suggest some correlation between these two events.

Hi-C analysis for normally cycling yeast cultures indicated several genomic interactions on chromosome 4, 7, 12, 13 and 15. A significant overlap was found for the genome proximity regions on chromosome 12 and the activated gene clusters (18 000 - 470 000). This region of chromosome 12 is an area with the highest intra-chromosomal interaction, and also include the rDNA genes. Not surprisingly, these genes are expressed as a cluster in this region.

The Hi-C data for the stationary phase cells exhibited no significant overlap for chromosome 12. Interestingly according to Rutledge *et al* rDNA expression is not significantly upregulated in stationary phase cells. (87) The absence of significant spatial contact in the same region of chromosome 12 during stationary phase is thus understandable.

Pyxis2 has thus clearly shown its utility in identifying gene clusters in terms of a specified functional genomic property, such as activated gene expression, and possible correlative biological signals. Here I demonstrated successfully that transcriptionally active clusters are in some cases associated with bound SAGA complex, but not with the epigenetic mark H3K9ac. This result suggests that H3K9ac is not a requirement for transcriptional activation of clusters of genes. I have also shown that there is an overlap in active gene clusters and regions of the genome that is spatially proximal in cycling yeast cells. This implies that expression of clusters of genes may dispose regions of the genome to assume a state of spatial proximity, although a comprehensive answer to this question will require additional analyses, looking at individual active genes and spatial proximity.

Pyxis 2 is thus clearly a useful bioinformatics tool and helpful addition to an arsenal of tools to investigate the phenomenon of clusters of genes and correlation with causative signals and is likely to help advancing insight into functional genomics.

References

1. Manuscript, A. (2009) NIH Public Access. **105**, 9–16
2. Haye, A., Albert, J., and Rooman, M. (2014) Modeling the Drosophila Gene Cluster Regulation Network for Muscle Development. 10.1371/journal.pone.0090285
3. Naseeb, S., and Delneri, D. (2012) Impact of Chromosomal Inversions on the Yeast DAL Cluster. **7**, 1–10
4. Roeder, R. G. (2019) expanding universe of factors and mechanisms. *Nat. Struct. Mol. Biol.* 10.1038/s41594-019-0287-x
5. Manuscript, A., and Pathways, S. (2014) NIH Public Access. **1829**, 361–375
6. Archambault, J., and Friesen, J. D. (1993) Genetics of Eukaryotic RNA Polymerases I , II , and III. **57**, 703–724
7. Manley, J. L., Fire, A., Cano, A., Sharp, P. A., and Gefter, M. L. (1980) of adenovirus genes. **77**, 3855–3859
8. Kadonaga, J. T. (2019) The transformation of the DNA template in RNA polymerase II transcription : a historical perspective. *Nat. Struct. Mol. Biol.* 10.1038/s41594-019-0278-y
9. Han, M., Kim, U., and Kayne, P. (1988) i : X. **7**, 2221–2228
10. Kadonaga, J. T. (2004) Regulation of RNA Polymerase II Transcription by Sequence-Specific DNA Binding Factors. **116**, 247–257
11. Croston, G. E., Kerrigan, L. A., Lira, L. M., Marshak, D. R., and Kadonaga, J. T. (1991) Sequence-Specific Antirepression Hi-Mediated Inhibition of Basal Polymerase II Transcription. 10.1126/science.1899487
12. Parker, C. S., and Roeder, R. G. (1977) Selective and accurate transcription of the Xenopus laevis 5S RNA genes in isolated chromatin by purified RNA polymerase III * Biochemistry : **74**, 44–48
13. Polymerase, P. R. N. A., Matsuis, T., Segallg, J., Weill, P. A., and Roederll, R. G. (1980) Multiple Factors Required for Accurate Initiation of Transcription by. **255**, 11992–11996
14. Mishima, Y., Kominami, R., and Muramatsu, M. (1982) Nucleic Acids Research. **10**, 6659–6670
15. Wood, R. (1996) The general transcription factors of RNA polymerase II
16. Drygin, D., Drygin, D., Rice, W. G., and Grummt, I. (2010) The RNA Polymerase I Transcription Machinery : An Emerging Target for the Treatment of Cancer The RNA Polymerase I Transcription Machinery : An Emerging Target for the Treatment of Cancer. 10.1146/annurev.pharmtox.010909.105844

17. Goodfellow, S. J., and Zomerdijk, J. C. B. M. (2013) Europe PMC Funders Group Basic Mechanisms in RNA Polymerase I Transcription of the Ribosomal RNA Genes. 10.1007/978-94-007-4525-4

18. Knutson, B. A., and Hahn, S. (2012) I general transcription factors. **333**, 1637–1640

19. Grummt, I., Mayer, C., Schmitz, K., Schmitt, N., Hoffmann-rohrer, U., and Scha, A. (2009) Article TAF12 Recruits Gadd45a and the Nucleotide Excision Repair Complex to the Promoter of rRNA Genes Leading to Active DNA Demethylation. 10.1016/j.molcel.2009.01.015

20. Nikolov, D. B., and Burley, S. K. (1997) RNA polymerase II transcription initiation : A structural view. **94**, 15–22

21. Abraham, K. J., Khosraviani, N., Chan, J. N. Y., Gorthi, A., Samman, A., Oshidari, R., Pietrobon, V., Patel, P. S., Algouneh, A., Singhania, R., Liu, Y., Yerlici, V. T., Carvalho, D. D. De, Ohh, M., Dickson, B. C., Hakem, R., Greenblatt, J. F., Lee, S., and Bishop, A. J. R. (2020) Nucleolar RNA polymerase II drives ribosome biogenesis. *Nature*. 10.1038/s41586-020-2497-0

22. Arimbasseri, A. G., Maraia, R. J., and Service, U. S. P. H. (2017) HHS Public Access. **41**, 546–559

23. Manuscript, A. (2013) NIH Public Access. **149**, 274–293

24. Bonhoure, N., Byrnes, A., Moir, R. D., Hodroj, W., Preitner, F., Praz, V., Marcelin, G., Jr, S. C. C., Martinez-lopez, N., Singh, R., Moullan, N., Auwerx, J., Willemin, G., Shah, H., Hartil, K., Vaitheesvaran, B., Kurland, I., Hernandez, N., and Willis, I. M. (2015) Loss of the RNA polymerase III repressor MAF1 confers obesity resistance. 10.1101/gad.258350.115.

25. Hahn, S. (2014) NIH Public Access machinery. 10.1038/nsmb763

26. Ranish, J. A., Yudkovsky, N., and Hahn, S. (1999) Intermediates in formation and activity of the RNA polymerase II preinitiation complex : holoenzyme recruitment and a postrecruitment role for the TATA box and TFIIB

27. Smale, S. T., and Kadonaga, J. T. (2003) PROMOTER ELEMENTS. 10.1146/annurev.biochem.72.121801.161520

28. Ohler, U., Liao, G., Niemann, H., and Rubin, G. M. (2002) Computational analysis of core promoters in the Drosophila genome

29. Driel, R. Van, Fransz, P. F., and Verschure, P. J. (2003) The eukaryotic genome : a system regulated at different hierarchical levels. 10.1242/jcs.00779

30. Lis, J. T. (2019) discovery in eukaryotic transcription regulation. *Nat. Struct. Mol. Biol.* 10.1038/s41594-019-0288-9

31. Vos, S. M., Farnung, L., Boehning, M., and Wigge, C. Structure of activated transcription complex Pol II-DSIF-PAF-

32. Pugh, B. F., and Venters, B. J. (2016) Genomic Organization of Human Transcription Initiation Complexes. 10.1371/journal.pone.0149339

33. Chodosh, L. A., Firell, A., Samuels, M., and Sharp, P. A. (1989) Inhibits Transcription Elongation by RNA Polymerase I1 in Vitro *. **264**, 2250–2257

34. Wu, C., Yamaguchi, Y., Benjamin, L. R., Horvat-gordon, M., Washinsky, J., Enerly, E., Larsson, J., Lambertsson, A., Handa, H., and Gilmour, D. (2003) NELF and DSIF cause promoter proximal pausing on the hsp70 promoter in Drosophila. 10.1101/gad.1091403.lation

35. Gilmour, D. S. (2014) NIH Public Access. **50**, 711–722

36. Muse, G. W., Gilchrist, D. A., Nechaev, S., Shah, R., Parker, J. S., Grissom, S. F., Zeitlinger, J., and Adelman, K. (2008) NIH Public Access. **39**, 1507–1511

37. Core, L. J., Waterfall, J. J., Gilchrist, D. A., Fargo, D. C., Adelman, K., and Lis, J. T. (2012) NIH Public Access. **2**, 1025–1035

38. Spellman, P. T., and Rubin, G. M. (2002) of Biology BioMed Central Evidence for large domains of similarly expressed genes in the Drosophila genome

39. Ridgway, P., Curie, I., and Cnrs, U. M. R. followed by the addition of two histone H2A-H2B dimers to form the core particle . The newly synthesized histones are specifically modified ; typically histone H4 is acetylated at Lys5 and Lys12 . During the maturation step ATP is required to establish a

40. Eberharter, A. Becker, P., (2002) Histone acetylation: a switch between repressive and permissive chromatin. *EMBO reports*, 3, 224-229.

41. Lachner, M., Jenuwein, T. (2002) The many faces of histone lysine methylation. *Current Opinion in Cell Biology*, 14, 286-298.

42. Zegerman, P., Canas, B., Pappin, D., Kouzarides, T. (2002) Histone H3 Lysine 4 Methylation Disrupts Binding of Nucleosome Remodeling and Deacetylase (NuRD) Repressor Complex. *Journal of Biological Chemistry*, 277, 11621-11624.

43. Selvi, R., Kundu, T. (2009) Reversible acetylation of chromatin: Implication in regulation of gene expression, disease and therapeutics. *Biotechnology Journal*, 4, 375-390.

44. Geiman, T., Robertson, K. (2002) Chromatin remodeling, histone modifications, and DNA methylation?how does it all fit together?. *Journal of Cellular Biochemistry*, 87, 117-125.

45. Ho, L., Crabtree, G. (2010) Chromatin remodelling during development. *Nature*, 463, 474-484.

46. Flaus, A., Owen-Hughes, T. (2004) Mechanisms for ATP-dependent chromatin remodelling: farewell to the tuna-can octamer? *Current Opinion in Genetics & Development*, 14, 165-173.

47. Venters, B., Pugh, B. (2009) How eukaryotic genes are transcribed. *Critical Reviews in Biochemistry and Molecular Biology*, 44, 117-141.

48. Andrulis, E., Neiman, A., Zappulla, D., Sternglanz, R. (1998) Erratum: Perinuclear localization of chromatin facilitates transcriptional silencing. *Nature*, 395, 525-525

49. Elizondo, L., Jafar-Nejad, P., Clewing, J., Boerkoel, C. (2009) Gene Clusters, Molecular Evolution and Disease: A Speculation. *Current Genomics*, 10, 64-75.

50. Oliver, B., Parisi, M., Clark, D. (2002) Journal search results - Cite This For Me. *Journal of Biology*, 1, 4

51. Price, M., Arkin, A., Alm, E. (2006) Journal search results - Cite This For Me. *BMC Bioinformatics*, 7, 19

52. Stoebel, D. (2004) Lack of Evidence for Horizontal Transfer of the lac Operon into Escherichia coli. *Molecular Biology and Evolution*, 22, 683-690.

53. Hurst, L., Pál, C., Lercher, M. (2004) The evolutionary dynamics of eukaryotic gene order. *Nature Reviews Genetics*, 5, 299-310.

54. Yi, G., Sze, S. Thon, M. (2007) Identifying clusters of functionally related genes in genomes. *Bioinformatics*, 23, 1053-1060.

55. Chang, C. F., Wai, K. M., & Patterton, H. G. 2004. Calculating the statistical significance of physical clusters of co-regulated genes in the genome: the role of chromatin in domain-wide gene regulation. *Nucleic acids research*, 32, 1798-1807

56. Bi, X., Braunstein, M., Shei, G., Broach, J. (1999) The yeast HML I silencer defines a heterochromatin domain boundary by directional establishment of silencing. *Proceedings of the National Academy of Sciences* 96, 11934-11939

57. Plath, K., Mlynarczyk-Evans, S., Nusinow, D., Panning, B. (2002) Xist RNA and the Mechanism of X Chromosome Inactivation. *Annual Review of Genetics* 36, 233-278

58. Litt, M. (2001) Correlation Between Histone Lysine Methylation and Developmental Changes at the Chicken beta -Globin Locus. *Science* 293, 2453-2455

59. Michalak, P. (2008) Coexpression, coregulation, and cofunctionality of neighboring genes in eukaryotic genomes. *Genomics* 91, 243-248

60. Barry, K., Stiles, J., Pietras, D., Melnick, L., and Sherman, F. (1987) Physical analysis of the COR region: a cluster of six genes in Saccharomyces cerevisiae. *Molecular and Cellular Biology* 7, 632-638

61. Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., and Davis, R. (1998) A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle. *Molecular Cell* 2, 65-73

62. Ford, H., and Pardee, A. (1999) Cancer and the cell cycle. *Journal of Cellular Biochemistry* 75, 166-172

63. Lercher, M. (2003) Coexpression of Neighboring Genes in Caenorhabditis Elegans Is Mostly Due to Operons and Duplicate Genes. *Genome Research* 13, 238-243

64. Birnbaum, K. (2003) A Gene Expression Map of the Arabidopsis Root. *Science* 302, 1956-1960

65. Williams, E. (2004) Coexpression of Neighboring Genes in the Genome of Arabidopsis thaliana. *Genome Research* 14, 1060-1067

66. Yang, Y., Cao, W., Wu, S., and Qian, W. (2017) Genetic Interaction Network as an Important Determinant of Gene Order in Genome Evolution. *Molecular Biology and Evolution* 34, 3254-3266

67. Pál, C., and Hurst, L. (2003) Evidence for co-evolution of gene order and recombination rate. *Nature Genetics* 33, 392-395

68. Sugino, R., and Innan, H. (2011) Natural Selection on Gene Order in the Genome Reorganization Process After Whole-Genome Duplication of Yeast. *Molecular Biology and Evolution* 29, 71-79

69. Wong, S., and Wolfe, K. (2005) Birth of a metabolic gene cluster in yeast by adaptive gene relocation. *Nature Genetics* 37, 777-782

70. Bumgamer, R. (2013) Overview of DNA Microarrays: Types, Applications, and Their Future. *Current protocols in molecular biology 22 10.1002/0471142727.mb2201s101*

71. Scholtens, D., von Heydebreck, A. (2005) Analysis of Differential Gene Expression Studies. In: Gentleman R., Carey V.J., Huber W., Irizarry R.A., Dudoit S. (eds) Bioinformatics and Computational Biology Solutions Using R and Bioconductor. *Statistics for Biology and Health. Springer,* New York, NY. https://doi.org/10.1007/0-387-29362-0_14

72. Obregón, F., Soto, P., Lavin, J., Cortázar, A., Barrio, R., Aransay, A., Cantera, R. (2018) Cluster Locator, online analysis and visualization of gene clustering. *Bioinformatics* 34, 3377-3379

73. Zhu, T., Liang, C., Meng, Z., Guo, S., and Zhang, R. (2017) GFF3sort: a novel tool to sort GFF3 files for tabix indexing. *BMC Bioinformatics* 18

74. Pal, K., Forcato, M., and Ferrari, F. (2018) Hi-C analysis: from data generation to integration. *Biophysical Reviews* 11, 67-78

75. Glueck, D., Mandel, J., Karimpour-Fard, A., Hunter, L., and Muller, K. (2008) Exact Calculations of Average Power for the Benjamini-Hochberg Procedure. *The International Journal of Biostatistics* 4

76. Quinlan, A., and Hall, I. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 26, 841-842

77. Freese, N., Norris, D., and Loraine, A. (2016) Integrated genome browser: visual analytics platform for genomics. *Bioinformatics* 32, 2089-2095

78. Raghupathy, N., and Durand, D. (2009) Gene Cluster Statistics with Gene Families. *Molecular Biology and Evolution* 26, 957-968

79. Herskowitz, I. (1988) Life cycle of the budding yeast Saccharomyces cerevisiae. *Microbiological Reviews* 52, 536-553

80. Nakato, R., and Sakata, T. (2020) Methods for ChIP-seq analysis: A practical workflow and advanced applications. *Methods*

81. Juicer and Juicebox for chromatin conformation analysis (2016) *Nature Methods* 13, 816-816

82. Eden, E., Navon, R., Steinfeld, I., Lipson, D., and Yakhini, Z. (2009) GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists. *BMC Bioinformatics* 10

83. Werner-Washburne, M. (2002) Comparative Analysis of Multiple Genome-Scale Data Sets. *Genome Research* 12, 1564-1573

84. Marcet-Houben, M., and Gabaldón, T. (2015) Beyond the Whole-Genome Duplication: Phylogenetic Evidence for an Ancient Interspecies Hybridization in the Baker's Yeast Lineage. *PLoS Biol* 13

85. Wolfe, K. (2015) Origin of the Yeast Whole-Genome Duplication. *PLoS Biol* 13

86. Eriksson, P., Ganguli, D., Nagarajavel, V., and Clark, D. (2012) Regulation of histone gene expression in budding yeast. *Genetics* 191, 7-20

87. Rutledge, M. T., Russo, M., Belton, J. M., Dekker, J., Broach, J. R. (2015) The yeast genome undergoes significant topological reorganization in quiescence. *Nucleic Acids Res* 43, 299-313

Addendum A

BED files of clusters formed during stationary phase exit

```
track name= Up15_clusters description="Item RGB demonstration"
itemRgb="On"
II    548362     558198     cluster    6.08E-03   0     0     0     250,0,0
IV    508147     543369     cluster    6.14E-03   0     0     0     250,0,0
IV    1149951    1170143    cluster    6.08E-03   0     0     0     250,0,0
X     188004     211577     cluster    8.36E-03   0     0     0     250,0,0
XII   582233     604787     cluster    8.03E-03   0     0     0     250,0,0
XVI   370978     393654     cluster    6.08E-03   0     0     0     250,0,0

track name= Up30_clusters description="Item RGB demonstration"
itemRgb="On"
I     68716 79435 cluster    3.50E-03   0     0     0     255,0,0
X     73787 76510 cluster    3.50E-03   0     0     0     255,0,0
XII   257992     263954     cluster    2.08E-03   0     0     0     255,0,0
XII   578362     604787     cluster    5.85E-03   0     0     0     250,0,0
XIV   493956     512396     cluster    3.50E-03   0     0     0     255,0,0
XVI   370978     393654     cluster    3.50E-03   0     0     0     255,0,0

track name= Up45_clusters description="Item RGB demonstration"
itemRgb="On"
I     65778 83227 cluster    4.74E-03   0     0     0     255,0,0
II    450881     483367     cluster    4.74E-03   0     0     0     255,0,0
IV    764178     771454     cluster    4.74E-03   0     0     0     255,0,0
VII   584895     617278     cluster    4.74E-03   0     0     0     255,0,0
X     188004     217309     cluster    6.78E-05   0     0     0     255,0,0
XII   257992     263954     cluster    4.09E-03   0     0     0     255,0,0
XIV   493956     512396     cluster    8.31E-03   0     0     0     250,0,0
XV    80348 94402 cluster    5.72E-03   0     0     0     250,0,0
XVI   370978     407539     cluster    9.57E-04   0     0     0     255,0,0

track name= Up60_clusters description="Item RGB demonstration"
itemRgb="On"
```

I	61316	83227	cluster	4.66E-03	0	0	255,0,0
II	450881	483367	cluster	1.48E-03	0	0	255,0,0
IV	765706	771454	cluster	9.88E-03	0	0	250,0,0
VII	584895	617278	cluster	9.60E-03	0	0	250,0,0
VIII	225525	237693	cluster	4.66E-03	0	0	255,0,0
X	188004	217309	cluster	1.48E-03	0	0	255,0,0
XI	549448	560652	cluster	4.66E-03	0	0	255,0,0
XII	228908	263954	cluster	3.61E-03	0	0	255,0,0
XIII	550206	558524	cluster	4.66E-03	0	0	255,0,0
XIV	493956	512396	cluster	9.88E-03	0	0	250,0,0
XVI	370978	407539	cluster	1.48E-03	0	0	255,0,0

Addendum B

```
import numpy as np
from decimal import *
class Combinations():

    def __init__(self):
        self.s = 0
    def Factorial(self, start, end):
        last_loop = []
        result = Decimal(1)
        for i in range(1,end+1):
            result = result*i
        return(result)

    def Combinations(self, n, r):
        C_n_r = self.Factorial(1,(n-r))
        C_r = self.Factorial(1,(r))
        C_n = self.Factorial(1,(n))
        result = Decimal(C_n/(C_n_r*C_r))
        return(result)

    def ClusterSignificance(self, total_length, total_responsive_genes,
window_size, genes_in_window):
        window_combinations =
         np.longdouble(self.Combinations(window_size, genes_in_window))

        chromosome_combinations =
         np.longdouble(self.Combinations(total_length,
         total_responsive_genes))
        combinations_of_balls_outside_window =
         np.longdouble(self.Combinations((total_length -
         window_size),(total_responsive_genes - genes_in_window)))
        result =
         np.longdouble(combinations_of_balls_outside_window *
         (window_combinations/chromosome_combinations))
        return(result)
```

```
import re
from itertools import groupby
import argparse
parser = argparse.ArgumentParser(description='Pyxis detects clusters of
regulated genes and returns the confidence')
parser.add_argument('-v', metavar='v', type=str, help="GFF file") #GFF
file
parser.add_argument('-s', metavar='s', type=str, help="List of regulated
genes") #file with upregulated genes
parser.add_argument('-w', metavar='w', type=str, nargs='?',
default='None', const='None', help='Statistical test used to detect
clusters')
parser.add_argument('-i', metavar='i', type=str, nargs='?',
default='ID=gene:', const='ID=gene:', help='Search string used to
identify Genes in GFF file (default: ID=gene:)')
parser.add_argument('-k', metavar='k', type=str, nargs='?',
default='Name=', const='Name=', help='Search string used to identify Gene
name in GFF file (default: Name=)')
parser.add_argument('-p', metavar='p', type=int, nargs='?', default=0.01,
const=0.01, help='P-value used to make statistical test more rigorous
(default: 0.05)')
parser.add_argument('-l', metavar='l', type=int, nargs='?', default=5,
const=5, help='Minimum window to detect gene clusters (default: 5)')
args = parser.parse_args()
GFF_file = args.v
files = args.s
idstring = args.i
sig_value = args.p
minimum_window = args.l
class GFF_parser():
    def __init__(self):
        self.chromosome_list = []
        self.sequence_region = []
```

```
    def chromosome(self,GFF_file):
        seq = ["##sequence-region"]
        # attribute that are used to identify the lines that    contain
the name of genes
        lines = []
        gff_file = open(GFF_file,"r")
        for line in gff_file:
            for att in seq:
                if att in line:
                    lines = line.split()
                    self.chromosome_list.append(lines[1])
        # finds lines in the opened files that  contains the
         lines with attribute
        #split the lines into strings and appends it into list  called
"lines"
        #from every list with appended string append the first
         element(string) to empty list "chromosome_list"
        return(self.chromosome_list)

    def sequence_region(self,GFF_file):
        seq = ["##sequence-region"]
        # attribute that are used to identify the lines that    contain
the name of genes
        lines = []
        gff_file = (GFF_file,"r")
        for line in gff_file:
            for att in seq:
                if att in line:
                    lines = line.split()
                    self.sequence_region.append(lines[2:4])
                    self.sequence_region = [tuple(int(x) for x in
                tup) for tup in self.sequence_region]
        # finds lines in the opened files that contains the lines
         with attribute
        #split the lines into strings and appends it into list  called
"lines"
```

```
        #append second to third element to empty list
         sequence_region
        #join every two elements into tuple and change the strings
         into integers
        return(self.sequence_region)

    def gene_region(self,chromosome_list,GFF_file):
        lines = []
        count = len(chromosome_list)
        gene_region = [[] for i in range(0,count)]
        #creates empty list of lists in range of count
        att = [args.i]
        gff_file = open(GFF_file,"r")
        for line in gff_file:
            for name in att:
                if name in line:
                    words = line.split()
                    if words[2] == "gene":
                        if args.k in words[8]:
                            lines.append(words)
                        else:
                            if args.k == "":
                                lines.append(words)
        header = [list(g) for k, g in groupby(lines, key=lambda x:
         x[0])]
        #create list of lists called header and group each lines
         into lists based on the first element (chromosome number)     of
the split lines
        genes = [[x[3:5] for x in y] for y in header]
        #create new list of lists called genes and append the    third
and fourth element from each split lines of nested    list header
        for t in range(count):
            while t < count:
                gene_region[t] = [tuple(int(x) for x in tup) for
                                  tup in genes[t]]
        #create paired elements from list genes and create tuples     and
convert elements(strings) to integers
```

```
                pass
                t = t+1
        return(gene_region)

    def gene_names(self, chromosome_list, GFF_file):
        lines = []
        att = [args.i]
        gff_file = open(GFF_file,"r")
        for line in gff_file:
            for name in att:
                if name in line:
                    words = line.split()
                    if words[2] == "gene":
                        #if third element contains the name gene
                        if args.k in words[8]:
                            lines.append(words)
                        else:
                            if args.k == "":
                                lines.append(words)

        #append the split lines to empty list called lines
        names = []
        gene_names = []

        header = [list(g) for k, g in groupby(lines, key=lambda x:
              x[0])]
        #create list of lists called header and group each lines
         into lists based on the first element (chromosome number)     of
the split lines
        for lines in header:
            for lists in lines:
                for element in lists:
                    if args.i in element:
                        words = [lists[0] ,(re.split(r'(:+|;+|=)',
                                    element))]
        #based on the characters you want to use to split the line
         containing "gene name"
```

```
                        names.append(words)
        for elements in names:
            #print(elements[1]) #- print this line and adjust   where
the gene name is located in the split line for  your GFF file
            word = [elements[0],elements[1][4]]
        #GFF files differ based on the line containing the gene  name so
the x might differ in elements[1][x](elements[1][x-   position of gene
name])
            gene_names.append(word)
        gene_names = [list(g) for k, g in groupby(gene_names,
                  key=lambda x: x[0])]

        gene_names = [[x[1] for x in y] for y in gene_names]

        gene_names = [list(g) for k, g in groupby(gene_names,
         key=lambda x: x[0])]

        gene_names = [[x[1] for x in y] for y in gene_names]

        return(gene_names)
```

```
#class that are imported into python
class HPArrayOfGeneInfo():

    def __init__(self):
        self.up_gene = []


    def GenesInRange(self, ball_positions, number_balls,
         minimum_window):
        start = 0
        begin = start

        while start < number_balls - 1 :
            try:
```

```
                if ball_positions[start+1] - ball_positions[start]
            <= minimum_window:
                    self.up_gene.append(ball_positions[start])
                    while ball_positions[start+1] -
                     ball_positions[start] <= minimum_window:
                      self.up_gene.append(ball_positions[start+1])
                        pass
                        start =start + 1
                else:
                    begin = start + 1

            except IndexError:
                break

            pass
            start = start + 1


        inds= [0]+[ind for ind,(i,j) in
                enumerate(zip(self.up_gene,self.up_gene[1:]),1) if    j-
i>5]+[len(self.up_gene)+1]
        up_genes = [self.up_gene[i:j] for i,j in
                     zip(inds,inds[1:])]
        #split the elements in list up_gene if they are bigger
         than 5 to create list of lists up_genes
        #Each list represents a "cluster"
        return(up_genes)


import numpy as np

class HPCombinations():

    def __init__(self):
        self.s = 0
    def Factorial(self, start, end):
        result = np.longdouble(1)
```

```
        for i in range(1,end+1):
            result = result*i
        return(result)

    def Combinations(self, n, r):
        C_n_r = self.Factorial(1,(n-r))
        C_r = self.Factorial(1,(r))
        C_n = self.Factorial(1,(n))
        result = np.longdouble(C_n/(C_n_r*C_r))
        return(result)

    def ClusterSignificance(self, total_length, total_responsive_genes,
window_size, genes_in_window):
        window_combinations =
         np.longdouble(self.Combinations(window_size, genes_in_window))

        chromosome_combinations =
         np.longdouble(self.Combinations(total_length,
         total_responsive_genes))
        combinations_of_balls_outside_window =
         np.longdouble(self.Combinations((total_length -
         window_size),(total_responsive_genes - genes_in_window)))
        result =     np.longdouble(combinations_of_balls_outside_window
*        (window_combinations/chromosome_combinations))
        return(result)
```

```
class HPFindClusters():
    #detects the number of clusters
    def __init__(self):
        self.number_of_clusters = 0

    def FindClusters(self, total_length, ball_positions,   number_balls,
minimum_window):
        self.number_of_clusters = 0
        start = 0
        begin = start
```

```
        cluster_number = 0
        count = 0
        while start < number_balls - 1 :
            if ball_positions[start+1] - ball_positions[start] <=
           minimum_window:
                if start == begin:
                    cluster_number = cluster_number +1
                    self.number_of_clusters = cluster_number
                else:
                    self.number_of_clusters = cluster_number
            else:
                begin = start + 1
            pass
            start = start + 1
        return(self.number_of_clusters)
```

```
class patterns():

    def __init__(self):
        self.s = 0

    def chrom(self, gene_names, gene):
        count = len(gene_names)
        #creates integer that represents number of    chromosomes(counts
number of lists in gene names)
        patternofclusters = [[] for i in range(0, count)]
        #creates empty nested list "patternofclusters" according      to
the number of chromosomes

        for number in range(0,count):
            for i in range(len(gene_names[number])):
                i = 0
                patternofclusters[number].append(i)
        #append 0's to nested list patternofclusters that  represents
balls(genes) on a stick(chromosome)
            for name in gene:
```

```
                #gene represents the list of upregulated genes so     for
element in the list of upregulated genes
                try:
                    position = gene_names[number].index(name)
                    #creates variable that finds index for each
        upregulated gene in nested list gene_names
                    patternofclusters[number][position] = 1
                    #add integer 1 to index in nested list
        patternofclusters
                except ValueError:
                    #if there is any index errors because gene is
not found in gene names let code continue
                      continue
        return(patternofclusters)
        #return the nested list patternofclusters
```

```
# coding: utf-8

from GFF_parser import GFF_parser
from HPArrayOfGeneInfo import HPArrayOfGeneInfo
from HPCombinations import HPCombinations
from HPFindClusters import HPFindClusters
from Combination import Combinations
from Patterns import patterns
from statsmodels.sandbox.stats.multicomp import multipletests

import csv
import argparse
import numpy as np
import itertools
import matplotlib.pyplot as plt
import re

parser = argparse.ArgumentParser(description='Pyxis detects clusters of
```

```
regulated genes and returns the confidence')
parser.add_argument('-v', metavar='v', type=str, help="GFF file") #GFF
file
parser.add_argument('-s', metavar='s', type=str, help="List of regulated
genes") #file with upregulated genes
parser.add_argument('-w', metavar='w', type=str, nargs='?',
default='None', const='None', help='Statistical test used to detect
clusters')
parser.add_argument('-i', metavar='i', type=str, nargs='?',
default='ID=gene:', const='ID=gene:', help='Search string used to
identify Genes in GFF file (default: ID=gene:)')
parser.add_argument('-k', metavar='k', type=str, nargs='?',
default='Name=', const='Name=', help='Search string used to identify Gene
name in GFF file (default: Name=)')
parser.add_argument('-p', metavar='p', type=float, nargs='?',
default=0.01, const=0.01, help='Lower P-value used to make statistical
test more rigorous (default: 0.01)')
parser.add_argument('-q', metavar='q', type=float,
nargs='?',default=0.05, const=0.05, help='Maximum p-value to make
statistical test more rigorous (default:0.05)')
parser.add_argument('-l', metavar='l', type=int, nargs='?', default=5,
const=5, help='Minimum window to detect gene clusters (default: 5)')
args = parser.parse_args()
print("The search string for the GFF file is: ", args.i)
print("The database string for the GFF file is: ", args.k)
print("The cut-off p-value is: ", args.p)
print("The minimum window for the detection of gene clusters is: ",
args.l)
GFF_file = args.v
files = args.s
idstring = args.i
sig_value = args.p
minimum_window = args.l



#files = file that contains the names of upregulated genes
##LTC
```

```
#creates accumulator list called gene
#files = file that contains the names of upregulated genes
##LTC
gene = []
#creates accumulator list called gene
files = open(files,"r")
for line in files:
    #for line in file
    lines = line.split()
    #split each line in the file
    gene.append("".join(lines))
    #append the split lines to a list called gene

gff_parser = GFF_parser()
chromosome_list = gff_parser.chromosome(GFF_file)
gene_names = gff_parser.gene_names(chromosome_list,GFF_file)
clusters = patterns()
patternofclusters = clusters.chrom(gene_names,gene)

##LTC

bed_data = []
hg_data = []
significant_genes = []
unsignificant_genes = []
g = []
bd = []
data = []
w = []
j = []

counter = -1

##LTC
for chromosome in range(len(chromosome_list)):
```

```
print("_____
_____")
    print("chromosome:",chromosome_list[chromosome])
    findclusters = HPFindClusters()
    #initialize the cluster called HPFindClusters
    combinations = HPCombinations()
    combination = Combinations()
    #initialize the cluster called HPCombinations
    ball_positions = [i for i, x in
     enumerate(patternofclusters[chromosome]) if x == 1]
    #append the index of the upregulated genes to a list called ball
positions
    print(ball_positions)
    number_balls = len(ball_positions)
    #number of balls represents the number of upregulated genes       per
chromosome
    print("number balls:",number_balls)
    total_length = len(gene_names[chromosome])
    #total length represents the number of genes per chromosome
    print("total length:",total_length)
    #number of gene distance which upregulated genes must be to be
considered a cluster
    number_of_clusters =
     findclusters.FindClusters(total_length,ball_positions,number_b
     alls,minimum_window)
    print("number of clusters:",number_of_clusters)
    if number_of_clusters >= 1:


         print("==================================================
         ==========================================")
        #if there is a gene cluster present on chromosome
        chromosome_length = total_length
        #chromosome length is equal to total number of genes on
         chromosome
        print("chromosome length:", chromosome_length)
        total_genes = number_balls
        #number of responsive genes on chromosome
```

```
        print("total genes:",total_genes)
        genes_in_range = HPArrayOfGeneInfo()
        cluster =
         genes_in_range.GenesInRange(ball_positions,number_balls,min
         imum_window)
        #cluster represents list of list that contains the index      of
clustered responsive genes that are within minimum    window:5
        print("cluster:",cluster)

        for window in cluster:
            counter += 1
            genes = []
            names = []
            print("gene cluster:",window)
            size = window[-1] - window[0] + 1
            #size represents the total number of genes in gene   window.
            print("window size:",size)
            genes_in_window = len(window)
            #genes_in_window represents the number of responsive
             genes in the window
            print("genes in window:",genes_in_window)
            for position in window:
                genes.append(gene_names[chromosome][position])
                bed_data.append(gene_names[chromosome][position])
                gene = GFF_parser()
                gene_region =
                 gene.gene_region(chromosome_list,GFF_file)
                names.append(gene_region[chromosome][position])
                if chromosome_length < 1000:
                    significance =
                       combinations.ClusterSignificance(chromosome_le
                       ngth, total_genes, size, genes_in_window)
                else:
                    significance =
                      combination.ClusterSignificance(chromosome_leng  th,
total_genes, size, genes_in_window)
                hg_data.append(significance)
```

```
            data.append(significance)

            words = (names[0][0],names[-1][1])
            g.append(words)

            print("gene names:",genes)
            for i in bed_data:
                lk =  [chromosome_list[chromosome],words[0],words[1],data
                 [counter]]
            bd.append(lk)

            bs = []
            r = []
            print("significance:","{:.2E}".format(significance))
if args.w == "None":
    sig_data = list(zip(bed_data, hg_data))
    print(sig_data)
    print(hg_data)
    lis = list(zip(bd,data))

    k = [list(i) for i in lis]
    for i in k:
        lin = [i[0][0],i[0][1],i[0][2],i[0][3],i[1]]
        r.append(lin)
        #print(r)

else:
    p_adjusted = multipletests(data, method=args.w)
    s = list(p_adjusted[1])
              # print(s)
    lis = list(zip(bd, s))
    #print(lis)
    k = [list(i) for i in lis]
    for i in k:
        lin = [i[0][0],i[0][1],i[0][2],i[0][3],i[1]]
        r.append(lin)
        #print(r)
```

```
l = []
if args.k == "":
    with open(GFF_file, "r") as f:
        for line in f:
            if str(args.i) in line:
                for i in bed_data:
                    o = line.split()
                    print("o",o)
                    p = o[8]

                    j = [(re.split(r'(:+|;+|=)', p))]
                    print(j)
                    for y in j:
                        #print(y[3])
                        if y[4] == i:
                            q =
                            [o[0],int(o[3]),int(o[4]),o[8]]
                            #print(q)
                            l.append(q)
else:
    with open(GFF_file, "r") as f:
        for line in f:
            if str(args.k) in line:
                if str(args.i) in line:
                    for i in bed_data:
                        o = line.split()
                        print("o",o)
                        p = o[8]

                        j = [(re.split(r'(:+|;+|=)', p))]
                        print(j)
                        for y in j:
                        #print(y[3])
                            if y[4] == i:
                                q =    [o[0],int(o[3]),int(o[4]),o
                                       [8]]
```

```
                                #print(q)
                                    l.append(q)

    t = []
    print(l)
    print(r)
    for i in l:
        for k in r:
            if i[0] == k[0] and i[1] == k[1]:
                u = [i[0],i[1],i[2],i[3],k[3],k[4]]
                t.append(u)

            if i[0] == k[0] and i[2] == k[2]:
                u = [i[0],i[1],i[2],i[3],k[3],k[4]]
                t.append(u)

            if i[0] == k[0] and i[1] > k[1] and i[2] < k[2]:
                u = [i[0],i[1],i[2],i[3],k[3],k[4]]
                t.append(u)


    y = []
    for i in t:
        p = [i[5]]
        y.append(p)
        flattened = [val for sublist in y for val in sublist]
        sig_data = list(zip(bed_data, flattened))
print("sig_data",sig_data)

if args.w == "None":
    print("_____________________________________________________________
    _________________")



else:
```

```
    print("_____________________________________________________________
    _________________")
    print("Correction test:",args.w)
    for i in r:
        print("chromosome:",i[0])
        print("region:",i[1],"-",i[2])
        print("corrected p-value:",i[4])

print("_____________________________________________________________________
______")

print("=====================================================================
============================")




def flatten(list):
  for o in list:
    for r in o:
      yield r


def myround(x, base=10):
    return base * round(x/base)

def bed_builder(GFF_file,sig_data):
    fl = 'track name= %s description="Item RGB demonstration"
itemRgb="On"' %args.s
    filename = "%s_genes.bed" %args.s
    k = open(filename,"w")
    k.write(fl + '\n')
    k.close()
    k = open(filename,"a+")
    BED = []

    for m in sig_data:
```

```
        if m[1] <= args.p:
            colour = str("255,0,0")
            if args.k == "":
                with open(GFF_file, "r") as f:
                    for line in f:
                        if str(args.i) in line:
                            o = line.split()
                        #print(o)
                            p = o[8]
                            j = [(re.split(r'(:+|;+|=)', p))]

                            for y in j:
                                if y[4] == m[0]:
                                #print(line)
                                    lines =    [np.array(str.split(lin
                                               e))[[0,3,4]],[m[0]],["{
                                               :.2E}".format(m[1])],[0
                                               ],[0],[0],[colour]]
                            #print(lines)
                                    new_lines = flatten(lines)
                                    BED.append(new_lines)
                            #print("p",p)
            else:
                with open(GFF_file, "r") as f:
                    for line in f:
                        if str(args.k) in line:
                            if str(args.i) in line:
                                o = line.split()
                        #print(o)
                                p = o[8]
                                j = [(re.split(r'(:+|;+|=)', p))]

                                for y in j:
                                    if y[4] == m[0]:
                                #print(line)
                                        lines =
                                                    [np.array(str.split
```

```
                                                (line))[[0,3,4]],[m
                                                [0]],["{:.2E}".form
                                                at(m[1])],[0],[0],[
                                                0],[colour]]
                            #print(lines)
                                        new_lines = flatten(lines)
                                        BED.append(new_lines)
                                    #print("p",p)

        if m[1] > args.p and m[1] < args.q:
            if args.k == "":
                with open(GFF_file, "r") as f:
                    for line in f:
                        if str(args.i) in line:
                            o = line.split()
                        #print(o)
                            p = o[8]
                            j = [(re.split(r'(:+|;+|=)', p))]

                            for y in j:
                                if y[4] == m[0]:
                                    x =     (np.log(m[1])+2.4740649358
                                            032383)/-
                                            0.007670633343976754
                                    if x >= 250:
                                        colour = str("250,0,0")
                                        lines =     [np.array(str.spli
                                                    t(line))[[0,3,4]],
                                                    [m[0]],["{:.2E}".f
                                                    ormat(m[1])],[0],[
                                                    0],[0],[colour]]
                                        new_lines = flatten(lines)
                                        BED.append(new_lines)
                                    if x <250:
                                        x = myround(x,10)
                                        colour = [int(x),0,0]
                                        c1 = ','.join(map(str,   colour))
```

```
                                    lines =    [np.array(str.split(
                                               line))[[0,3,4]],[m[0
                                               ]],["{:.2E}".format(
                                               m[1])],[0],[0],[0],[
                                               c1]]
                                    new_lines = flatten(lines)
                                    BED.append(new_lines)
        else:
            with open(GFF_file, "r") as f:
                for line in f:
                    if str(args.k) in line:
                        if str(args.i) in line:
                            o = line.split()
                    #print(o)
                            p = o[8]
                            j = [(re.split(r'(:+|;+|=)', p))]

                            for y in j:
                                if y[4] == m[0]:
                                    x =       (np.log(m[1])+2.47406
                                              49358032383)/-
                                              0.007670633343976754
                                    if x >= 250:
                                        colour =
                                                      str("250,0,0")
                                        lines =       [np.array(str
                                                      .split(line))
                                                      [[0,3,4]],[m[
                                                      0]],["{:.2E}"
                                                      .format(m[1])
                                                      ],[0],[0],[0]
                                                      ,[colour]]
                                        new_lines =   flatten(lines)
                                        BED.append(new_lines)
                                    if x <250:
                                        x = myround(x,10)
                                        colour = [int(x),0,0]
```

```
                                            c1 = ','.join(map(str,
colour))
                                            lines =     [np.array(str.s
                                                        plit(line))[[0,
                                                        3,4]],[m[0]],["
                                                        {:.2E}".format(
                                                        m[1])],[0],[0],
                                                        [0],[c1]]
                                            new_lines =
                                                        flatten(lines)
                                            BED.append(new_lines)
        if m[1] > args.q:
            continue
    bedbuilder = csv.writer(k, delimiter='\t')
    bedbuilder.writerows(BED)

    return(bedbuilder)

bedbuilder = bed_builder(GFF_file, sig_data)

def clusters(r):
    fl = 'track name= %s_clusters description="Item RGB demonstration"
itemRgb="On"' %args.s
    filename = "%s_clusters.bed" %args.s
    k = open(filename,"w")
    k.write(fl + '\n')
    k.close()
    k = open(filename,"a+")
    BD = []
    if args.w == "None":
        for i in r:
            if i[3] <= args.p:
                colour = str("255,0,0")
                lin =
                        [i[0],i[1],i[2],"cluster","{:.2E}".format(i[
                        3]),0,0,0,colour]
                BD.append(lin)
```

```
            if i[3] > args.p and i[3] < args.q:
                x = (np.log(i[4])+2.4740649358032383)/-
                0.007670633343976754

                if x >= 250:
                    colour = str("250,0,0")
                    lin =   [i[0],i[1],i[2],"cluster","{:.2E}".format
                            (i[3]),0,0,0,colour]
                    BD.append(lin)
                if x < 250:
                    x = myround(x,10)
                    colour = [int(x),0,0]
                    c1 = ','.join(map(str, colour))
                    lin =   [i[0],i[1],i[2],"cluster","{:.2E}".format
                            (i[3]),0,0,0,c1]
                    BD.append(lin)
            if i[3] > args.q:
                continue
    else:
        for i in r:
            if i[4] <= args.p:
                colour = str("255,0,0")
                lin =      [i[0],i[1],i[2],"cluster","{:.2E}".format(i[
                        4]),0,0,0,colour]
                BD.append(lin)

            if i[4] > args.p and i[4] < args.q:
                x = (np.log(i[4])+2.4740649358032383)/-
                     0.007670633343976754

                if x >= 250:
                    colour = str("250,0,0")
                    lin =    [i[0],i[1],i[2],"cluster","{:.2E}".forma
                            t(i[4]),0,0,0,colour]
                    BD.append(lin)
                if x < 250:
```

```
                    x = myround(x,10)
                    colour = [int(x),0,0]
                    c1 = ','.join(map(str, colour))
                    lin =     [i[0],i[1],i[2],"cluster","{:.2E}".form
                              at(i[4]),0,0,0,c1]
                    BD.append(lin)
            if i[4] > args.q:
                continue
    bedbuilder = csv.writer(k, delimiter='\t')
    bedbuilder.writerows(BD)
    return(bedbuilder)


clusters(r)
```

Addendum C

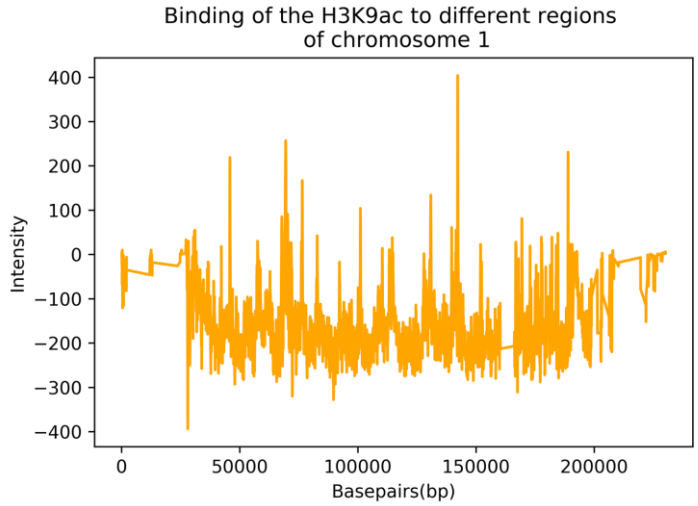


Figure 1:CHIP-seq analysis for H3K9ac chr1

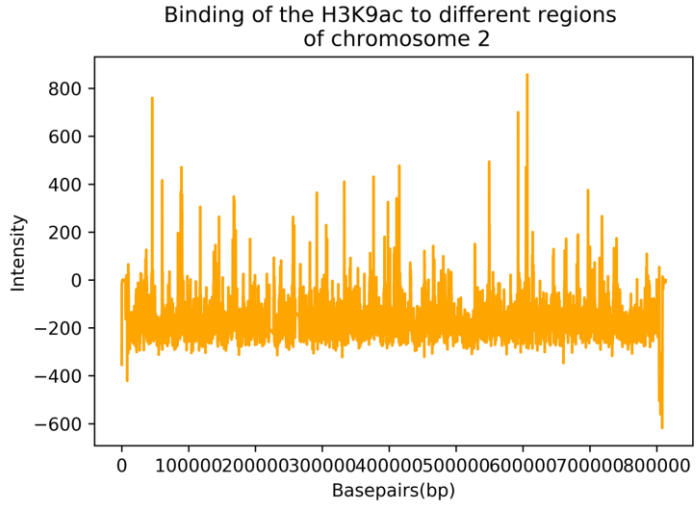


Figure 2: CHIP-seq analysis H3K9ac chr2

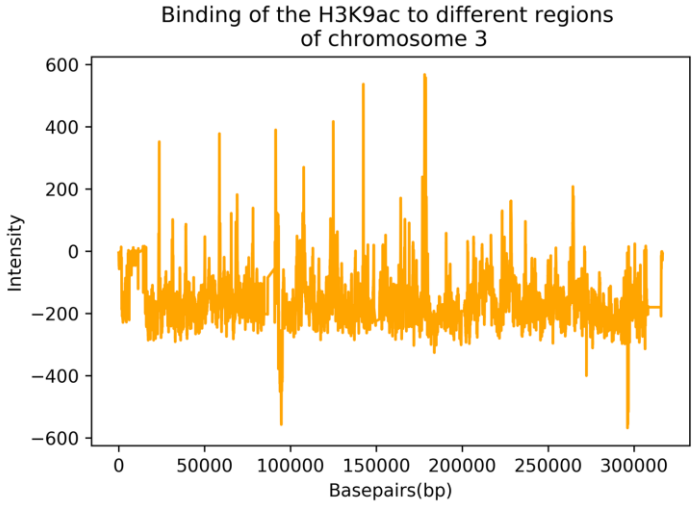


Figure 3: CHIP-seq analysis H3K9ac chr3

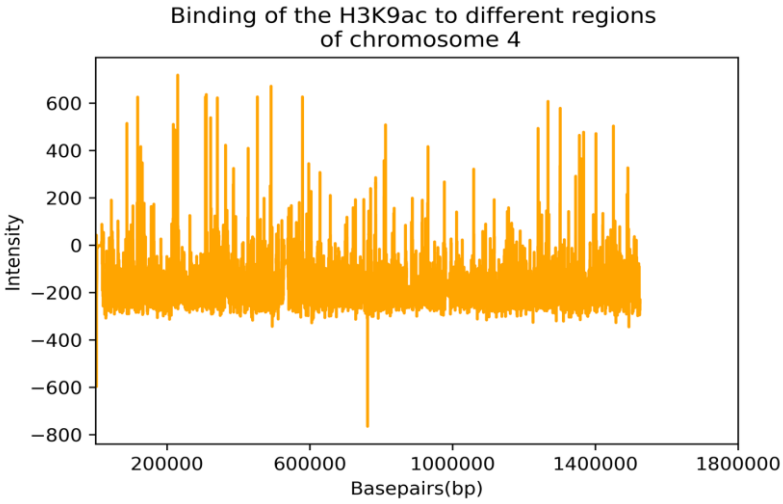


Figure 4: CHIP-seq analysis H3K9ac chr4

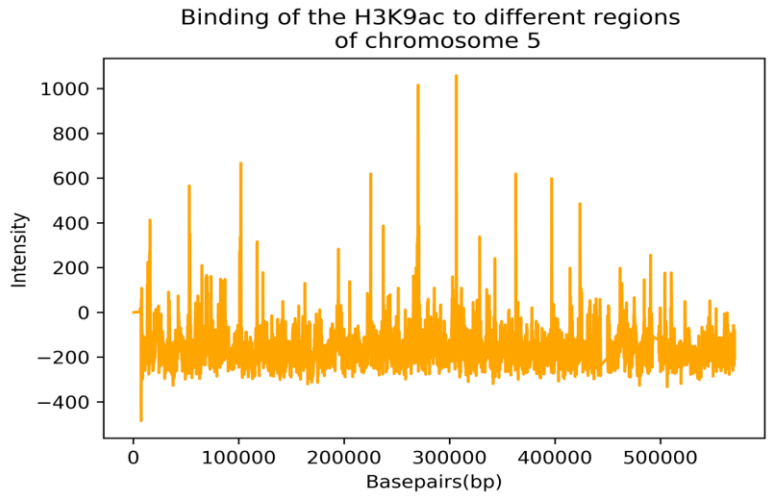


Figure 5: CHIP-seq analysis H3K9ac chr5

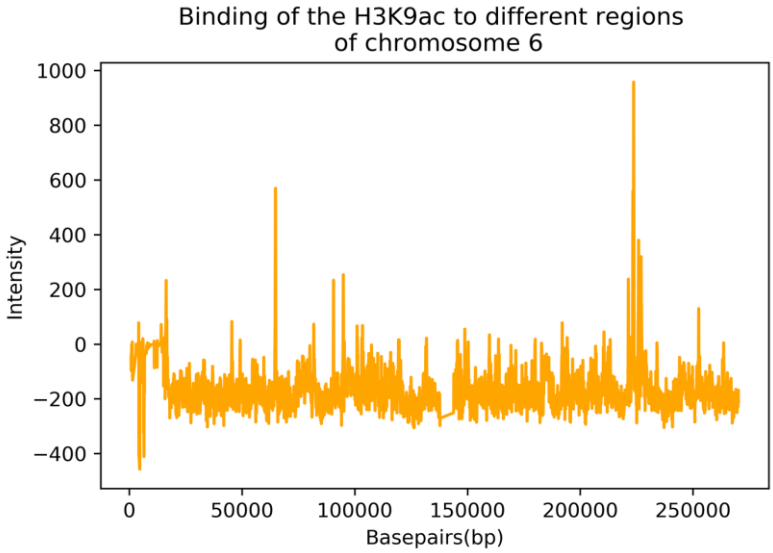


Figure 6: CHIP-seq analysis H3K9ac chr6

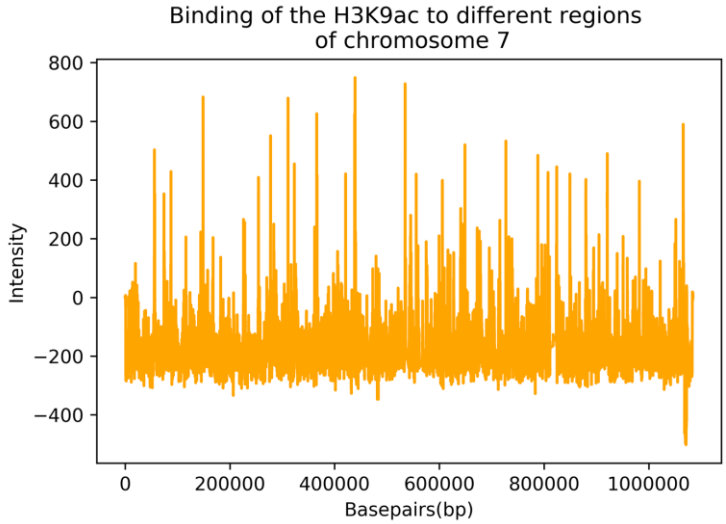


Figure 7: CHIP-seq analysis H3K9ac chr7

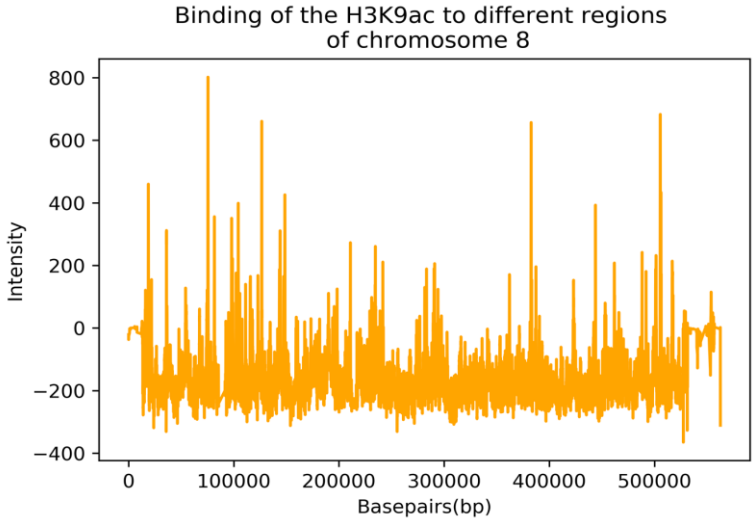


Figure 8: CHIP-seq analysis H3K9ac chr8

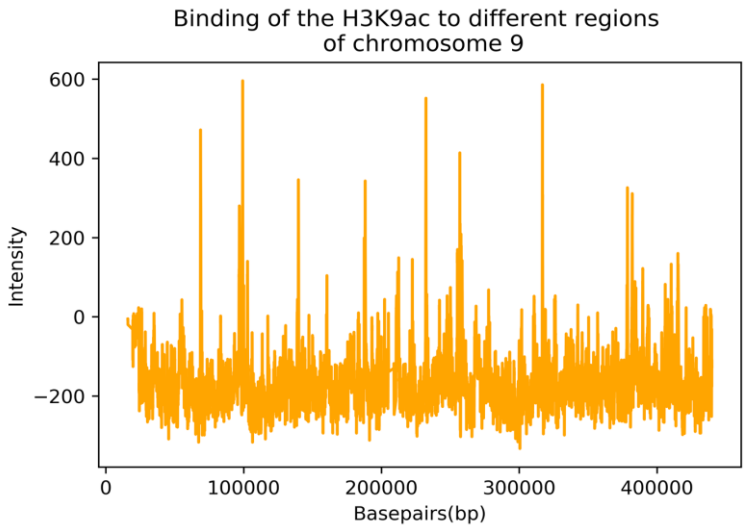


Figure 9: CHIP-seq analysis H3K9ac chr9

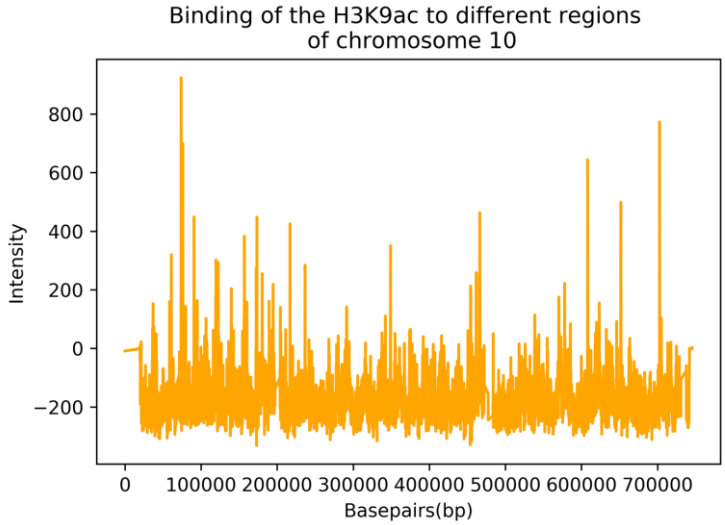


Figure 10: CHIP-seq analysis H3K9ac chr10

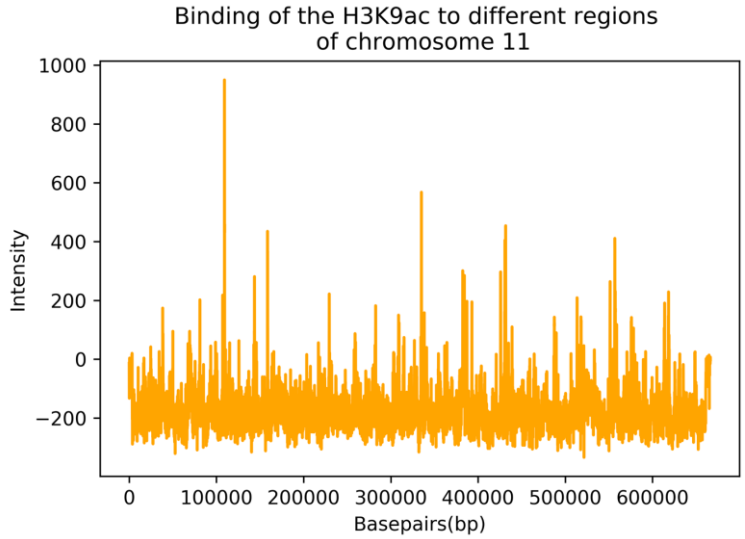


Figure 11: CHIP-seq analysis H3K9ac chr11

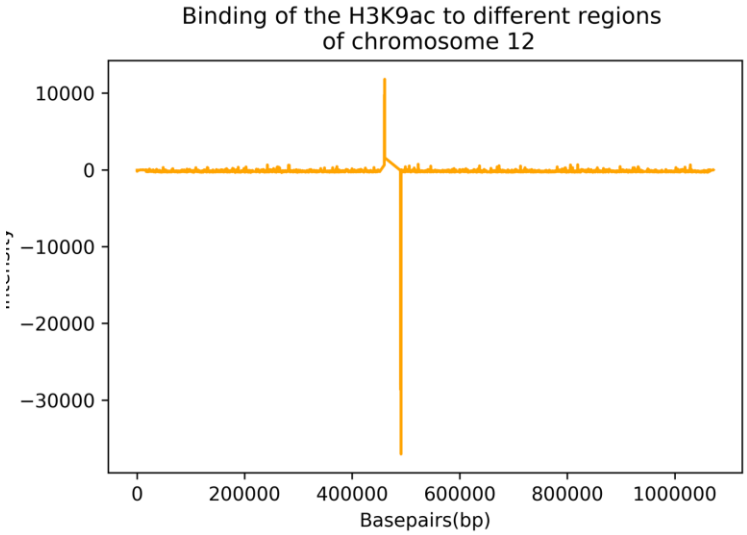


Figure 12: CHIP-seq analysis H3K9ac chr12

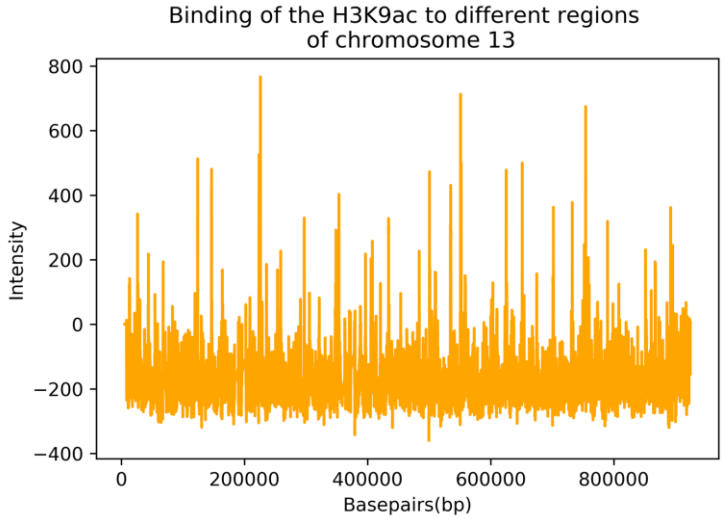


Figure 13: CHIP-seq analysis H3K9ac chr13

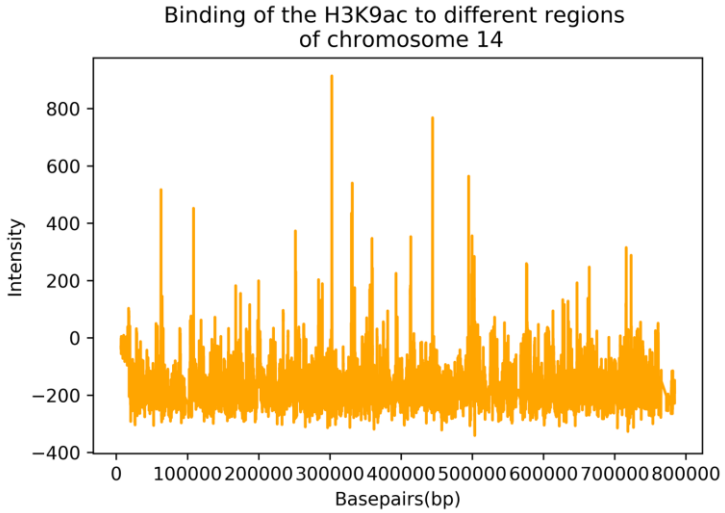


Figure 14: CHIP-seq analysis H3K9ac chr14

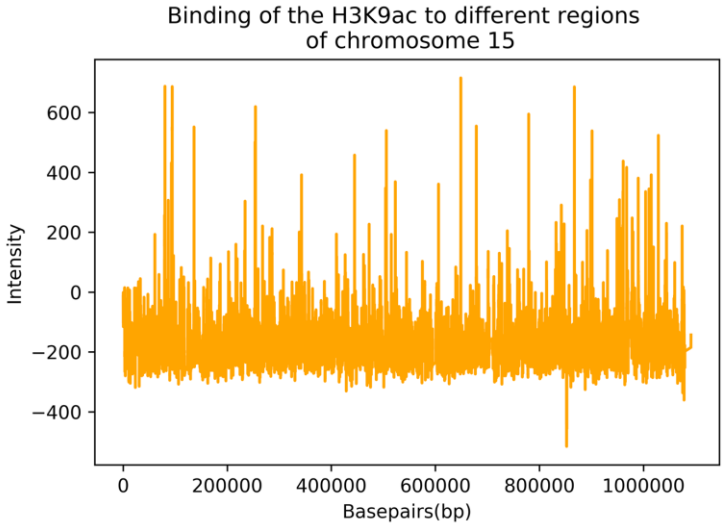


Figure 15: CHIP-seq analysis H3K9ac chr15

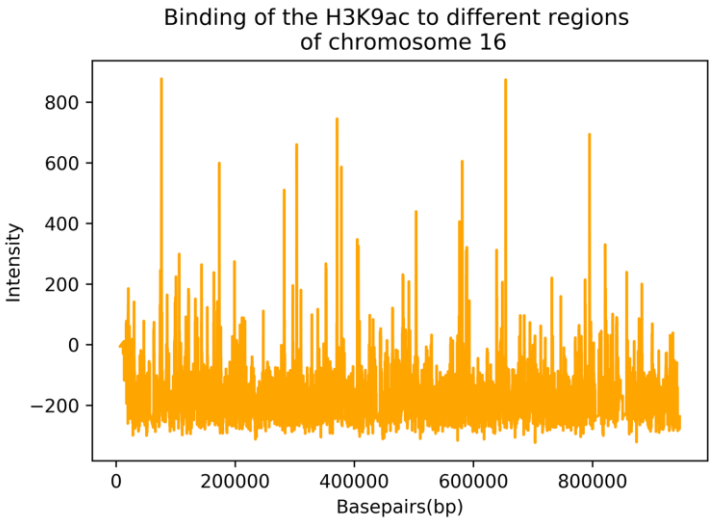


Figure 16: CHIP-seq analysis H3K9ac chr16

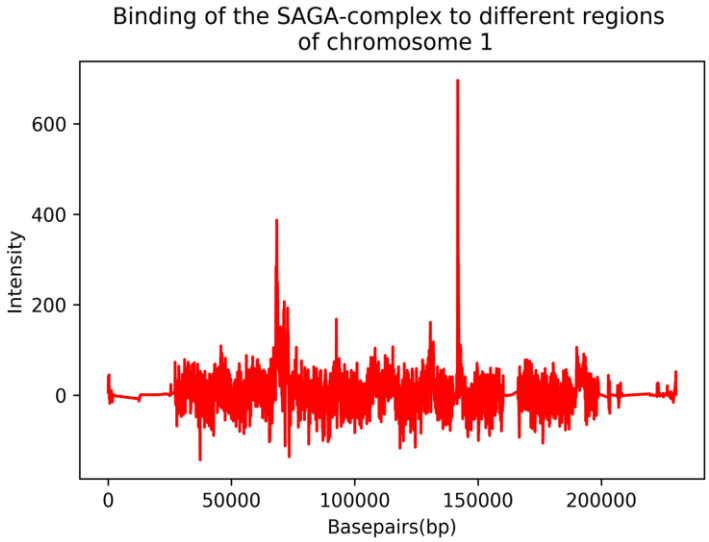


Figure 17: CHIP-seq analysis SAGA chr1

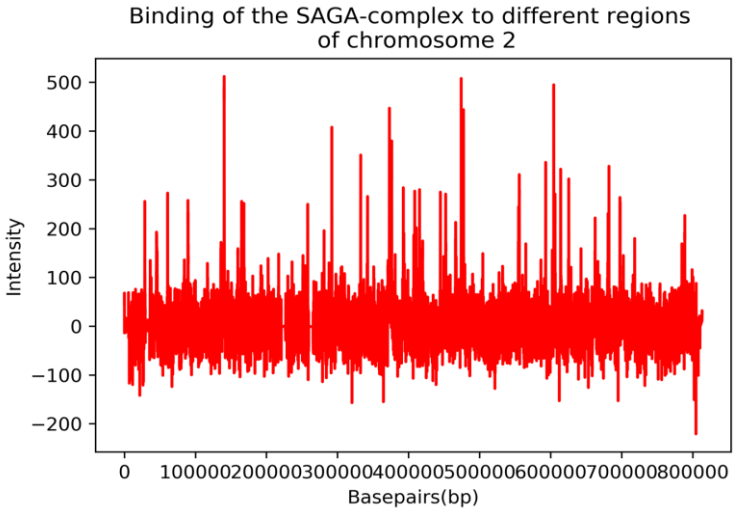


Figure 18: CHIP-seq analysis SAG1A chr2

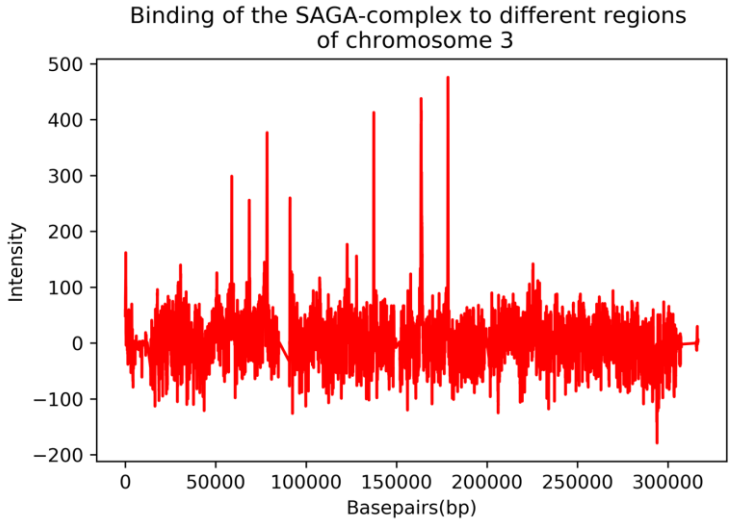


Figure 19: CHIP-seq analysis SAGA chr3

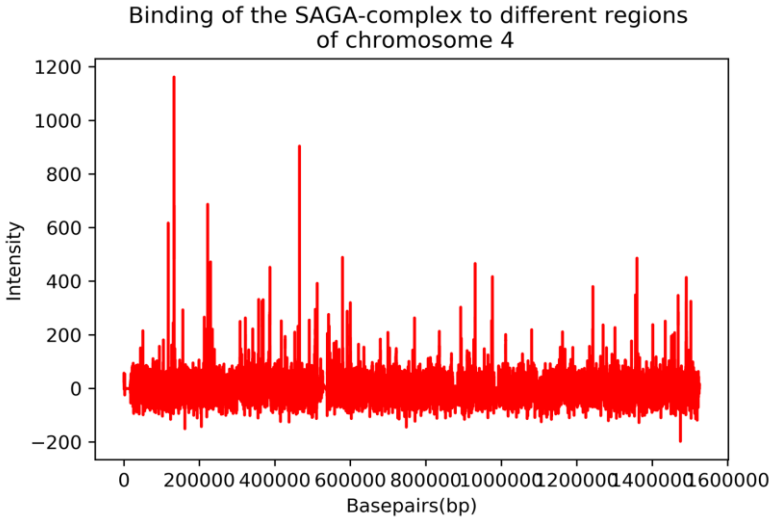


Figure 20: CHIP-seq analysis SAGA chr4

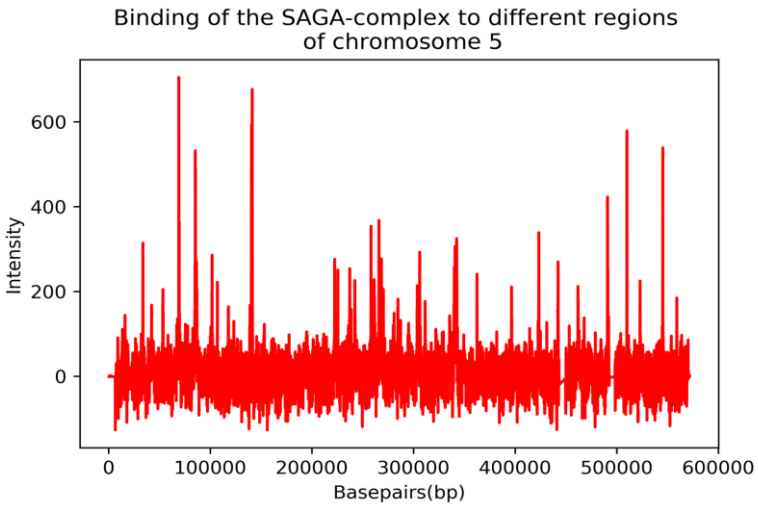


Figure 21: CHIP-seq analysis SAGA chr5

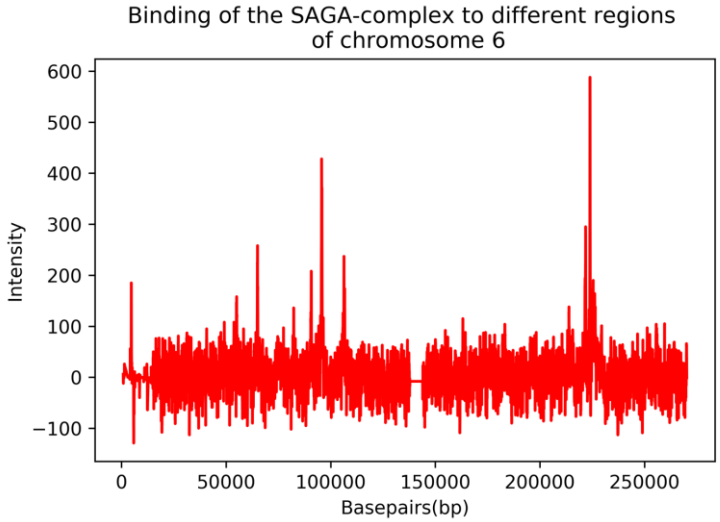


Figure 22: CHIP-seq analysis SAGA chr6

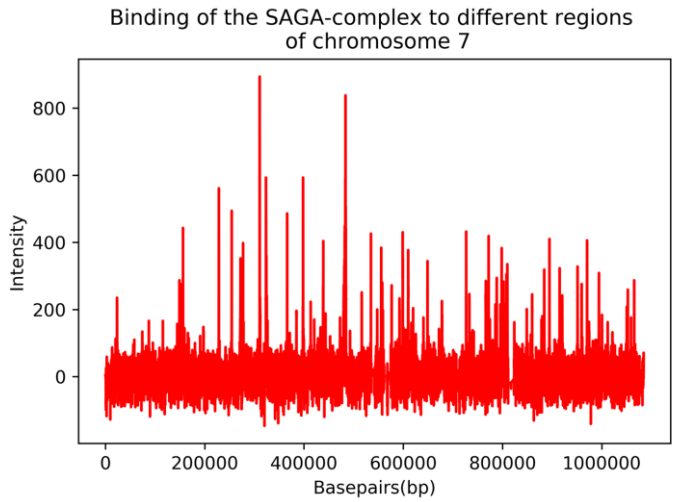


Figure 23: CHIP-seq analysis SAGA chr7

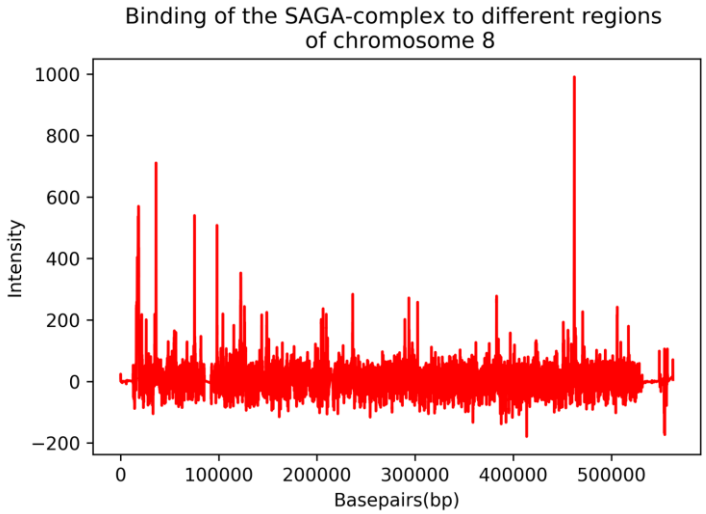


Figure 24: CHIP-seq analysis SAGA chr8

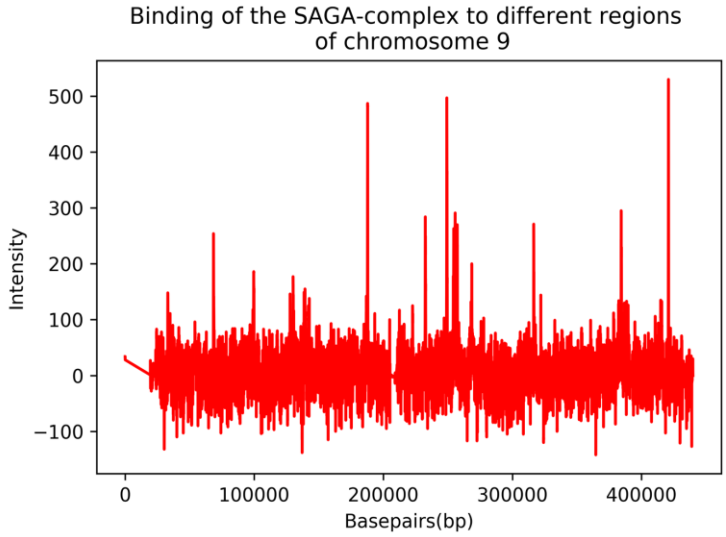


Figure 25: CHIP-seq analysis SAGA chr9

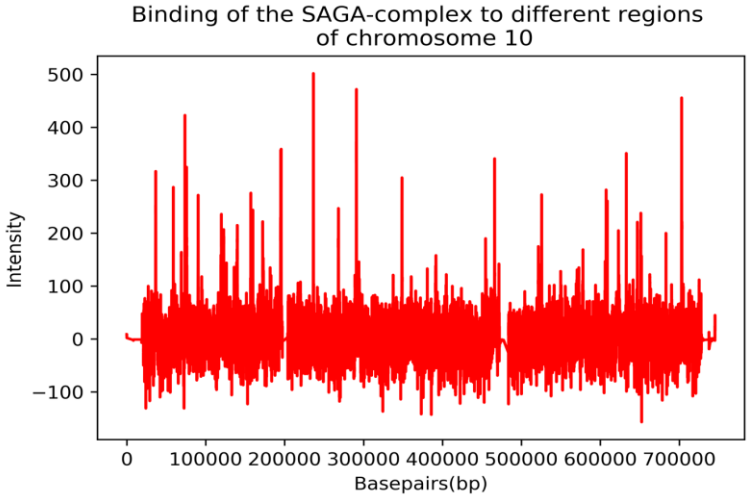


Figure 26: CHIP-seq analysis SAGA chr10

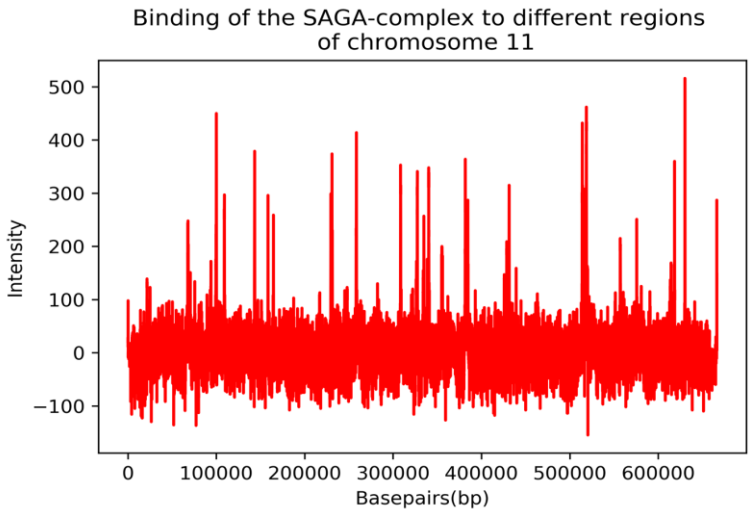


Figure 27: CHIP-seq analysis SAGA chr11

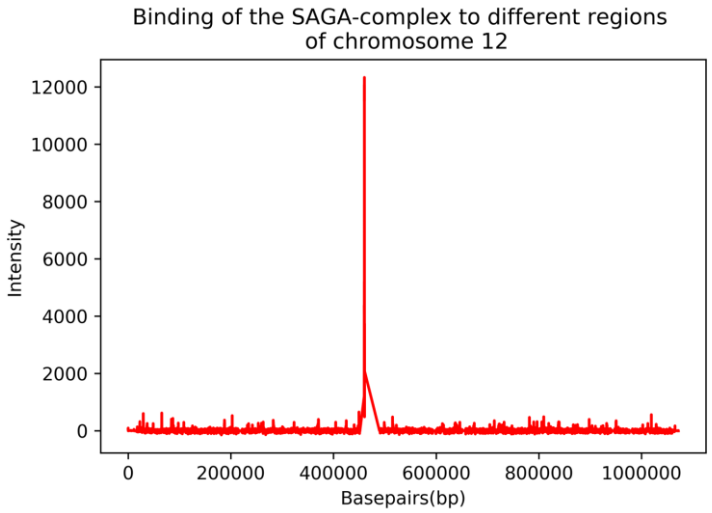


Figure 28: CHIP-seq analysis SAGA chr12

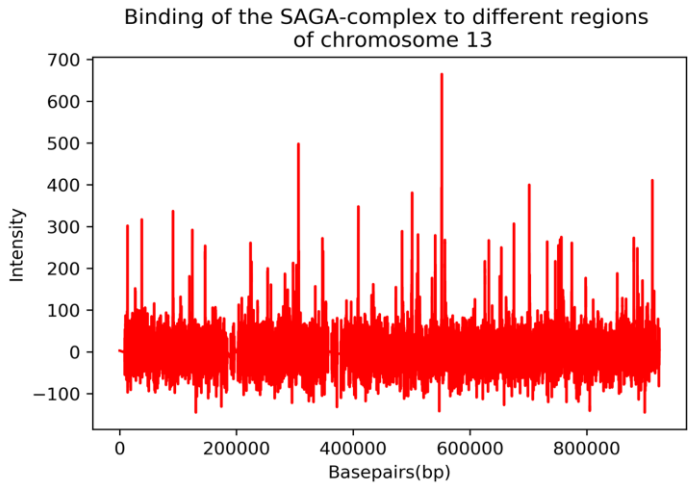


Figure 29: CHIP-seq analysis SAGA chr13

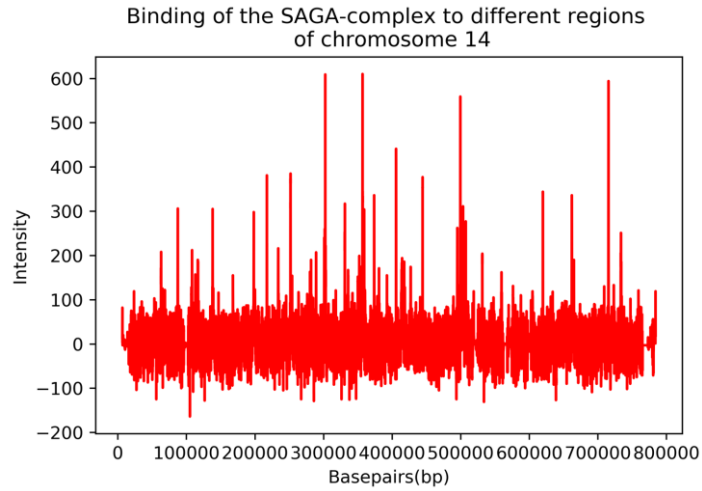


Figure 30: CHIP-seq analysis SAGA chr14

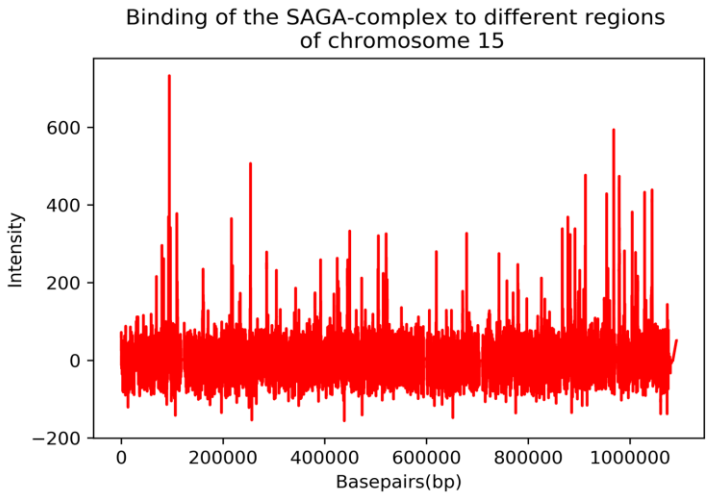


Figure 31: CHIP-seq analysis SAGA chr15

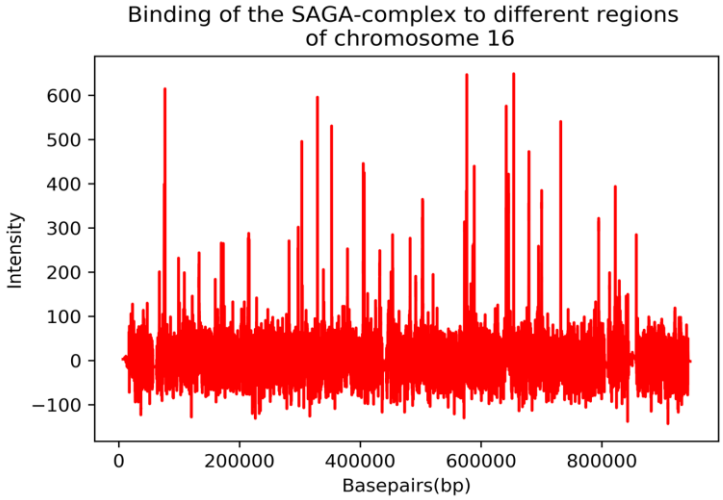


Figure 32: CHIP-seq analysis SAGA chr16